Optimización del proceso ETL en Big Data de la empresa “Casa de Incentivos Casintour‟ mediante técnicas de paralelismo basado en procesos durante el año 2023

Peña Chunez, Kevin Daniel y Simbaña Llumiquinga, Mirian Estefanía

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Trabajo de titulación, previo a la obtención del título de Ingeniero/a en Tecnologías de la Información

Ing. Guayasamín Guanga, César Adrián, Mgtr

01 de febrero del 2023

Copyleaks

Plagiarism and AI Content Detection Report

Trabajo_de_titulación_Peña_Chunez_...

Scan details

Scan time:	Total Pages:	Total Words:
April 10th, 2024 at 19:50 UTC	67	16641

Plagiarism Detection

8.4%

Types of plagiarism		Words
Identical	5.5%	914
Minor Changes	1.4%	231
Paraphrased	1.5%	245
Omitted Words	0%	0

AI Content Detection

9.8%

Text coverage		Words
AI text	9.8%	1639
Human text	90.2%	15002

Learn more

Plagiarism Results: (63)

Cordova_B-4_601514507_FSTopo.pdf 4.1%

https://data.fs.usda.gov/geodata/rastergateway/data3/60145/fstopo/cordova_b-4_601514507_fstopo.pdf

EEEEEE EEEEEE EEEEEE EEEEEE EEEEEE EEEEEE EE EE EEEEEEEEEEEEEEE EEEEEEEEEEEEEEE EEEEEEEEEEEEEEE EEEEEEEEEEEEEEE EEEEEEEEEEEEEEE...

File.ashx?id_org=9629&id_dokumenty=35627 2.2%

https://www.mb.net.cz/assets/file.ashx?id_org=9629&id_dokumenty=35627

C CC C C C CC C CC CC C CC C C C CC C CC CC C C C CC C C C C CC CC C C C C C C C C CC CC C C...

Firma

CESAR ADRIAN GUAYASAMIN GUANGA

Ing. Guayasamín Guanga, César Adrián, Mgtr

C. C 1715897680

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

Carrera de Tecnologías de la Información

Certificación

Certifico que el trabajo de integración curricular: **"Optimización del proceso ETL en Big Data de la empresa "Casa de Incentivos Casintour" mediante técnicas de paralelismo basado en procesos durante el año 2023"** fue realizado por los señores **Peña Chunez, Kevin Daniel y Simbaña Llumiquinga, Mirian Estefanía,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 27 de marzo del 2024

CESAR ADRIAN GUAYASAMIN GUANGA

Ing. Guayasamín Guanga, César Adrián, Mgtr

C. C 1715897680

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

Carrera de Tecnologías de la Información

Responsabilidad de Autoría

Nosotros, **Peña Chunez, Kevin Daniel y Simbaña Llumiquinga, Mirian Estefanía,** con cédulas de ciudadanía n° 0450038039 y n° 1727423962, declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **"Optimización del proceso ETL en Big Data de la empresa "Casa de Incentivos Casintour" mediante técnicas de paralelismo basado en procesos durante el año 2023"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, Sangolquí, 27 de marzo del 2024

Peña Chunez, Kevin Daniel
C.C.: 0450038039

Simbaña Llumiquinga, Mirian Estefanía
C.C.: 1727423962

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

Carrera de Tecnologías de la Información

Autorización de Publicación

Nosotros, **Peña Chunez, Kevin Daniel y Simbaña Llumiquinga, Mirian Estefanía,** con cédulas de ciudadanía n° 0450038039 y n° 1727423962 , autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **"Optimización del proceso ETL en Big Data de la empresa "Casa de Incentivos Casintour" mediante técnicas de paralelismo basado en procesos durante el año 2023"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 27 de marzo del 2024

Peña Chunez, Kevin Daniel
C.C.: 0450038039

Simbaña Llumiquinga, Mirian Estefanía
C.C.: 1727423962

Dedicatoria

A mis queridos padres Luis y Martha y a mi hermano Ronny,

Con profundo amor y gratitud, dedico este trabajo de titulación a ustedes, quienes han sido mi inspiración, apoyo incondicional y fuente de fortaleza a lo largo de este arduo camino académico. Su constante confianza en mí, sacrificio y aliento han sido los pilares que han sostenido mis sueños y aspiraciones.

A través de cada desafío y obstáculo, su guía y amor me han impulsado a perseguir la excelencia y superar mis propios límites. Su dedicación y sacrificio han sido el motor que me impulsa hacia el éxito, y por ello, hoy les dedico este logro con profunda gratitud y admiración.

Su hijo y hermano Kevin.

Dedicatoria

A mis queridos padres Homero y Mirian, fuente inagotable de amor y motivación, cuya inquebrantable dedicación ha sido mi faro en esta travesía académica. A ustedes les debo la fuerza que me impulsa a alcanzar cada meta, y es con gratitud infinita que les dedico este logro. Su apoyo constante, sus palabras alentadoras y su inquebrantable fe en mi capacidad han sido el cimiento sobre el cual he construido este trabajo de titulación. A través de las alegrías y desafíos, su respaldo incondicional ha sido mi mayor impulso. Gracias por ser mis guías, mis confidentes y mis mayores admiradores. Este logro no solo es mío, sino también de ustedes, quienes han sido mi apoyo inquebrantable. A mis padres, mi eterna gratitud y amor.

Su hija, Estefanía

Agradecimientos

Quisiera tomar un momento para expresar mi más sincero agradecimiento a todas las personas que han sido parte fundamental en el éxito de mi trabajo de titulación:

A mis padres Luis y Martha: Su esfuerzo, dedicación, preocupación y protección han sido la principal razón de este logro académico les agradezco el permitirme estudiar y su plena confianza en mí.

A mi hermano: Agradezco su constante guía y enseñanza a lo largo de mi vida gracias por ser el mejor ejemplo de persona que puedo tener.

A mi amada María José y a su familia: Tu amor, apoyo y comprensión han sido mi mayor motivación durante todo este proceso. Gracias por estar siempre a mi lado y por ser mi fuente constante de inspiración, de igual manera, agradezco a tu familia que ha sido un gran apoyo para mí.

A mi familia: Su incondicional apoyo y aliento han sido un pilar fundamental en mi camino hacia la culminación de este proyecto. Gracias por creer en mí y por brindarme todo su amor y comprensión.

A mi compañera de trabajo y hermana Estefi: Tu colaboración, consejos y amistad han sido invaluable. Gracias por tu ayuda y por compartir este viaje conmigo.

A mis amigos: Su ánimo, alegría y palabras de aliento han sido un impulso constante a lo largo de este proceso. Gracias por estar siempre ahí para mí y por ser parte de este logro.

A mi tutor: Su guía, conocimientos y dedicación han sido fundamentales en mi formación académica. Gracias por compartir su sabiduría y por inspirarme a alcanzar nuevas metas.

A todos ustedes, les estoy profundamente agradecido por su apoyo incondicional, por creer en mí y por ser parte de este importante capítulo en mi vida. Sin su ayuda, este logro no habría sido posible.

De Kevin Peña.

Agradecimientos

Agradezco, a Dios por darme la fuerza de voluntad necesaria para seguir adelante en cada paso. De manera especial a mis padres quienes son y serán siempre mi ejemplo a seguir, a mi padre, por las horas que pasó acompañándome mientras estudiaba, por cuidar de mí y aconsejarme; a mi madre, por sus palabras de aliento, comprensión y por siempre tener un abrazo para mí.

A mis abuelitos, tíos, primos y prima, que creyeron en mí y estuvieron prestos a brindarme su apoyo ante cualquier circunstancia, sin ellos no lo habría logrado. Agradezco a Pablo, mi compañero de vida, por su amor incondicional, por recordarme que puedo lograr lo que me proponga, por tomar mi mano y acompañarme en cada paso, siempre veré en él una fuente de inspiración. No puedo dejar de mencionar a mis amigos, Liliana, Avo y Elian, quienes han hecho de esta etapa universitaria, una oportunidad para compartir risas y experiencias amenas, con ellos confirmé que “Los amigos de la universidad te salvan un poco la carrera y otro poco la vida”, siempre ocuparán un lugar especial en mi corazón. A mi hermano Kevin, por ser exactamente eso, un hermano y un amigo invaluable, su amistad me ha motivado siempre. A mi tutor, por su guía y conocimiento en el desarrollo de este trabajo.

Gracias a todos.

Estefanía.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Este estudio resalta que la empresa “Casa de Incentivos Casintour” enfrenta el reto de procesar y analizar grandes volúmenes de datos mediante un proceso de extracción, transformación y carga (ETL) para llevarlos a un repositorio centralizado. A medida que la cantidad de datos aumenta, este proceso se vuelve más lento debido a su naturaleza secuencial, sin aprovechar el procesamiento paralelo disponible en entornos de Big Data.

La optimización del proceso ETL en Big Data es esencial para “Casa de Incentivos Casintour” debido a la ineficiencia de recursos al no aprovechar completamente el hardware, lo que puede requerir inversiones adicionales sin mejoras notables en el rendimiento y la pérdida de oportunidades comerciales al demorar la disponibilidad de los datos procesados, lo que limita la toma de decisiones basadas en datos en tiempo real, crucial en un entorno competitivo empresarial.

Además, este estudio destaca la necesidad de abordar la falta de paralelismo que afecta directamente el rendimiento, eficiencia, escalabilidad y oportunidades comerciales. Resolver este problema es fundamental para mejorar el procesamiento de datos, acelerar la disponibilidad de la información y aprovechar los recursos en entornos de Big Data. Para abordar los desafíos

mencionados y garantizar el éxito en la optimización del proceso ETL en “Casa de Incentivos Casintour”, se adoptará la metodología SCRUM para la ejecución de este proyecto de tesis.

Palabras clave: hilos de programación, procesos, rendimiento, iso 25010, paralelismo

Abstract

This study highlights that the company "Casa de Incentivos Casintour" faces the challenge of processing and analyzing large volumes of data through an extract, transform and load (ETL) process to bring it to a centralized repository. As the amount of data increases, this process becomes slower due to its sequential nature, without taking advantage of the parallel processing available in Big Data environments.

Optimizing the ETL process in Big Data is essential for the company "Casa de Incentivos Casintour” due to the inefficient use of resources in Big Data, due to the inefficiency of resources by not fully leveraging hardware, which may require additional investments without noticeable performance improvements and the loss of business opportunities by delaying the availability of processed data, which undermines real-time data-driven decision making, crucial in a competitive environment.

In addition, this study highlights the need to address the lack of parallelism that directly affects performance, efficiency, scalability and business opportunities. Addressing this issue is essential to improve data processing, accelerate information availability, and maximize resource utilization in the Big Data environment.

To address the above challenges and ensure success in optimizing the ETL process at "Casa de Incentivos Casintour", the agile SCRUM methodology will be adopted for the execution of this thesis project.

Key words: programming threads, processes, performance, iso 25010, parallelism.

Capítulo I

Introducción

En este capítulo se abordan los antecedentes, la problemática, la justificación, los objetivos, y el alcance de la investigación sobre la optimización del proceso ETL en Big Data de la empresa "Casa de Incentivos Casintour" durante el año 2023. Se exploran los eventos previos que condujeron a la necesidad de mejorar el proceso ETL y se destacan los desafíos específicos que enfrenta la empresa en este contexto.

La justificación se fundamenta en la importancia de mejorar la eficiencia en el manejo de datos, crucial en una era donde la velocidad y el volumen de la información son determinantes para la toma de decisiones empresariales. Además, se exponen las razones que respaldan la elección de técnicas de paralelismo basado en procesos como enfoque para la optimización.

Se detallan los objetivos de la investigación y se definen los logros específicos esperados con la implementación de estrategias de paralelismo. También se define con claridad el alcance del estudio, para lo cual se establecen los límites y alcances dentro de los cuales se llevará a cabo la investigación.

Adicionalmente, se llevará a cabo un estado del arte donde se revisarán tres artículos relevantes que aportan al tema, se explora su conexión con la optimización, el ETL y Big Data.

Esta revisión literaria contribuirá a contextualizar y fundamentar teóricamente la investigación propuesta.

Antecedentes

El aumento continuo en la cantidad de datos ha revelado una desaceleración progresiva en la eficiencia del proceso ETL de la empresa. Esta desaceleración se atribuye a la naturaleza secuencial del proceso, que no capitaliza completamente las capacidades de procesamiento disponibles en entornos de Big Data.

El proceso ETL en Maricus ha generado una subutilización significativa de los recursos de hardware. Actualmente, el procesamiento de boletos demora entre 2 y 3 horas, lo cual es una demora considerable en comparación con la necesidad de disponibilidad de datos en tiempo real. Esta ineficiencia no solo impacta la toma de decisiones basadas en datos, sino que también plantea la amenaza de inversiones adicionales sin mejoras sustanciales en el rendimiento.

Dada la creciente demanda y el potencial futuro de crecimiento de Maricus, la optimización de los recursos del computador en el que está alojado se vuelve esencial para evitar demoras en la disponibilidad de datos procesados, lo que podría afectar negativamente a la empresa y resultar en la pérdida de oportunidades comerciales. La implementación de paralelismo en el proceso ETL se presenta como una solución necesaria para agilizar el tiempo de procesamiento, mejorar la eficiencia, y preparar a Maricus para afrontar su crecimiento futuro de manera más efectiva.

Problemática

La compañía "Casa de Incentivos Casintour" se encuentra ante el desafío sustancial de gestionar y analizar vastos volúmenes de datos en su entorno empresarial. Para abordar esta tarea, se ha implementado un proceso de extracción, transformación y carga (ETL), el cual

implica la obtención de datos desde diversas fuentes, su posterior transformación a un formato compatible, y finalmente su carga en un repositorio centralizado.

No obstante, conforme la magnitud de los datos se incrementa, se ha observado una desaceleración progresiva y una disminución en la eficiencia del proceso ETL. Este fenómeno puede atribuirse a la naturaleza secuencial del procedimiento, en el cual cada etapa (extracción, transformación y carga) se ejecuta de manera lineal. Este enfoque no aprovecha plenamente las capacidades de procesamiento paralelo que están disponibles en los entornos de Big Data, lo que resulta en un rendimiento subóptimo.

Es imperativo abordar esta problemática para optimizar la eficiencia del proceso de manejo de datos, se deben considerar soluciones que permitan una ejecución más ágil y paralela de las tareas ETL. Esto no solo agilizará las operaciones internas, sino que también posibilitará una gestión más efectiva de los crecientes volúmenes de datos, de modo que se garantiza así una toma de decisiones informada y ágil para la empresa "Casa de Incentivos

La optimización del proceso ETL en el contexto de Big Data emerge como una necesidad apremiante para "Casa de Incentivos Casintour", y este imperativo se sustenta en múltiples razones esenciales.

En primer lugar, la falta de paralelismo en el proceso ETL resulta en una utilización ineficiente de los recursos de hardware disponibles, como procesadores y memoria. Esta subutilización conlleva a una asignación poco eficaz de recursos, lo que genera la posibilidad de tener que invertir en hardware adicional. Sin embargo, esta inversión no garantiza mejoras sustanciales en el rendimiento del proceso, lo que amplifica la importancia de optimizar este aspecto para aprovechar al máximo los recursos existentes y mejorar la eficiencia global del sistema.

Además, la demora en la disponibilidad de datos procesados representa una amenaza significativa para "Casa de Incentivos Casintour". La capacidad de tomar decisiones basadas en datos en tiempo real es crítica en un entorno empresarial competitivo. La falta de optimización en el proceso ETL puede llevar a la pérdida de oportunidades comerciales valiosas, ya que la toma de decisiones se ve obstaculizada por la indisponibilidad oportuna de información procesada. Esta situación compromete directamente la capacidad de la empresa para reaccionar ágilmente a oportunidades emergentes.

En resumen, la carencia de paralelismo en el proceso ETL durante el año 2023 se presenta como un desafío que afecta la eficiencia operativa, la escalabilidad y la capacidad de "Casa de Incentivos Casintour" para capitalizar oportunidades comerciales. La optimización de este proceso se vuelve imperativa para potenciar el rendimiento del sistema, acelerar la disponibilidad de información procesada y asegurar la agilidad necesaria en la toma de decisiones en un entorno empresarial dinámico y competitivo.

Objetivos

Objetivo General

Proponer una estrategia de implementación que aproveche el paralelismo basado en procesos para mejorar la eficiencia y el rendimiento del ETL en Big Data de la empresa “Casa de Incentivos Casintour" durante el año 2023.

Objetivos Específicos

- Investigar y comprender los conceptos y principios fundamentales del proceso ETL en entornos de Big Data, así como las técnicas y herramientas del paralelismo para la optimización de dicho proceso.

- Analizar en profundidad la infraestructura actual de procesamiento de datos en la empresa 'Casa de Incentivos Casintour', tanto la arquitectura de hardware como software utilizado para el proceso ETL en Big Data.
- Desarrollar un prototipo de la implementación propuesta, con tecnologías y herramientas adecuadas para habilitar el procesamiento paralelo de datos en ETL.
- Recopilar datos relevantes sobre el rendimiento, la escalabilidad y la eficiencia del proceso mejorado y compararlo con el proceso ETL tradicional para validar los resultados obtenidos y analizar el impacto de la optimización en términos de tiempo de ejecución, consumo de recursos y calidad de los datos procesados.

Alcance

El trabajo de titulación se enfocará en la optimización del proceso de Extracción, Transformación y Carga (ETL) en "Casa de Incentivos Casintour" mediante la integración de la norma ISO 25010, con especial énfasis en la dimensión de eficiencia de desempeño. El alcance incluirá un enfoque comparativo antes y después de la optimización, que aborda los siguientes aspectos:

Análisis Detallado del Proceso ETL Actual (Modelo Tradicional)

Identificación y documentación detallada de cada etapa del proceso ETL en el modelo tradicional.

Evaluación de las tecnologías y herramientas utilizadas actualmente en el proceso ETL.

Diagnóstico de Desafíos Actuales y Oportunidades de Mejora (Modelo Tradicional)

Identificación de los desafíos específicos que enfrenta el proceso ETL en términos de eficiencia, rendimiento y escalabilidad.

Evaluación de las oportunidades de mejora en base a los estándares y prácticas recomendadas de la norma ISO 25010.

Integración de la Norma ISO 25010 en el Proceso ETL (Modelo Tradicional)

Adaptación de los principios y criterios de la norma ISO 25010 al contexto del proceso ETL en el modelo tradicional.

Desarrollo de métricas específicas basadas en la ISO 25010 para evaluar la eficiencia de desempeño en el modelo tradicional.

Implementación de Mejoras Iterativas y Ágiles (Modelo Tradicional)

En el desarrollo continuo de mejoras para el proceso ETL, la metodología ágil SCRUM fue implementada, centrándose especialmente en la obtención minuciosa de requisitos que posibilitaron la elaboración detallada de historias de usuario.

Es crucial resaltar que la adopción de SCRUM se llevó a cabo con la finalidad específica de optimizar y agilizar la comunicación efectiva con el cliente, mediante lo cual se logró fortalecer así la eficiencia y la adaptabilidad en el marco del proyecto.

Evaluación Comparativa de Resultados (Modelo Tradicional vs. Mejorado)

Comparación de las métricas de eficiencia de desempeño antes y después de la implementación de mejoras bajo el modelo tradicional.

Análisis detallado de los beneficios obtenidos en ambos modelos.

Transición al Modelo Optimizado

Diseño de un plan detallado para la transición del modelo tradicional al modelo optimizado, con la incorporación de las mejores prácticas de la norma ISO 25010.

Este enfoque estratégico busca maximizar la eficiencia operativa y garantizar una

adaptación sin contratiempos, para asegurar así una transición efectiva hacia un sistema optimizado y conforme a los estándares de calidad establecidos.

Implementación del Modelo Optimizado

Ejecución de la transición planificada al modelo optimizado, con el empleo de las lecciones aprendidas del modelo tradicional. Además, se llevó a cabo un monitoreo continuo durante la implementación para asegurar una integración sin inconvenientes y ajustes efectivos en tiempo real.

Validación y Retroalimentación en Ambos Modelos

Obtención de retroalimentación por parte de los responsables del proceso ETL y otros stakeholders clave en ambos modelos (tradicional y optimizado).

Validación de los resultados obtenidos en ambos contextos y ajustes finales basados en la retroalimentación recibida.

Documentación y Presentación de la Tesis

Elaboración de un documento detallado que describa la metodología utilizada, los hallazgos, las implementaciones realizadas y los resultados obtenidos en ambos modelos.

Presentación formal de la tesis, en la que se destaque la importancia de la integración de la norma ISO 25010 en la optimización del proceso ETL y sus implicaciones para "Casa de Incentivos Casintour".

Este alcance ampliado permitirá una evaluación más completa de los beneficios de la optimización del proceso ETL en "Casa de Incentivos Casintour", al comparar directamente los resultados antes y después de la implementación de mejoras, lo cual proporcionará una visión clara de la eficacia de la intervención propuesta.

Definición de la Investigación

La investigación es un proceso sistemático y organizado que tiene como objetivo ampliar y generar conocimiento a partir de la indagación, la reflexión, la experimentación y la documentación de un asunto o problema en particular (Universidad de Lima, 2023). En el desarrollo de la investigación se utilizan fuentes de información de diversos autores, cuyas ideas se reconocen y se cita su autoría (Universidad de Lima, 2023). La investigación puede ser de diferentes tipos, como la investigación científica, la investigación de mercado, la investigación policial, entre otras (Cheesman de Rueda, 2010). La investigación se caracteriza por ser empírica, objetiva, organizada, coherente, analítica, rigurosa, verificable y sistemática (TDX, 2010). La información previa sobre el tema, problemática o fenómeno a estudiar es importante para el desarrollo de la investigación (Cheesman de Rueda, 2010). La definición de investigación puede variar según el contexto y la disciplina, pero en general, se refiere a la búsqueda de conocimiento y la verdad (Coelho, 2023).

Estado del Arte

En el marco actual del estado del arte, resulta esencial considerar tres estudios clave que abordan aspectos fundamentales en el ámbito del Extraer, transformar y cargar (ETL), optimización y Big Data. Estos estudios, cuidadosamente seleccionados, proporcionan una perspectiva integral y valiosa para enriquecer el presente trabajo de investigación.

Marco ETL de próxima generación para abordar los desafíos que plantean los Big Data

El crecimiento exponencial de los datos ha planteado nuevos desafíos para los marcos de trabajo ETL tradicionales en el contexto de Big Data. Este artículo aborda el problema de la inadecuación de los marcos de trabajo ETL convencionales para enfrentar los retos relacionados con la variedad, el volumen y la velocidad de los datos en Big Data. Se destaca que los marcos de trabajo existentes fueron diseñados para crear Data Warehouses

tradicionales, los cuales son eficientes en el procesamiento de cálculos ligeros en conjuntos de datos más pequeños. Sin embargo, el Big Data requiere una nueva generación de informática avanzada que no procese estos procesos.

Para resolver este problema, se propone un marco ETL escalable para ayudar a los desarrolladores a ampliar las operaciones ETL y agregar operaciones para Big Data. Este marco de trabajo consta de varios componentes clave: el componente de User-Defined Functions (UDFs), el recomendador, el modelo de costos y el agente de monitoreo.

El componente de UDFs proporciona plantillas de código paralelizable para los operadores de Big Data más utilizados y PASs (Parallelizable Analytical Service) genéricos. Esto permite aprovechar el paralelismo de tareas y el paralelismo de datos para mejorar el rendimiento de la ejecución de los flujos de trabajo ETL. Por otro lado, el recomendador utiliza algoritmos de aprendizaje automático para optimizar los flujos de trabajo ETL basándose en los metadatos recopilados durante las ejecuciones anteriores. Esto garantiza una optimización continua y adaptativa de los flujos de trabajo en función de los patrones identificados en los datos.

El modelo de costos proporciona una base para el análisis y la optimización de los costos monetarios y de rendimiento asociados con la ejecución de los flujos de trabajo ETL. Esto permite tomar decisiones informadas sobre cómo optimizar tanto el costo como el rendimiento de las ejecuciones. Finalmente, el agente de monitoreo permite supervisar las ejecuciones de los flujos de trabajo ETL, identificar cuellos de botella de rendimiento, informar errores, programar ejecuciones y recopilar estadísticas de rendimiento esenciales.

Juntos, estos sistemas ETL escalables brindan soluciones integrales para resolver los desafíos que se presentan con Big Data al mejorar la eficiencia y efectividad de los flujos de

trabajo ETL. El enfoque en el paralelismo, la optimización basada en aprendizaje automático y la consideración de los costos y el rendimiento contribuyen a una ejecución más efectiva y eficiente de los flujos de trabajo ETL en el entorno de Big Data (Ali, 2018).

Mejores prácticas de ETL (extraer, transformar, cargar)

Este estudio analiza las mejores prácticas para optimizar las operaciones ETL y se centra en estrategias que mejoran el rendimiento, la eficiencia y la calidad de los procesos. Se destaca la importancia de seguir estas prácticas para garantizar la integridad de los datos, la reducción de errores y la mejora de la eficiencia operativa.

El estudio proporciona una serie de soluciones y enfoques para abordar los desafíos asociados con los trabajos de ETL. Entre las soluciones propuestas se encuentran la optimización del proceso de ETL, la limpieza de datos, la detección y manejo de errores, la partición de datos, la indexación, el almacenamiento en caché, la reutilización de funciones y objetos comunes, la automatización, el procesamiento paralelo, la gestión de recursos y la monitorización y seguimiento de los trabajos de ETL.

Se enfatizó la importancia del procesamiento paralelo como una estrategia efectiva para mejorar el rendimiento de los trabajos de ETL al aprovechar recursos adicionales. Esta técnica permite distribuir la carga de trabajo en múltiples procesadores o nodos, lo que resulta en una reducción significativa en el tiempo de procesamiento y un aumento en la eficiencia global.

Además de las soluciones técnicas, el artículo también resalta la importancia de la documentación, la colaboración y la comunicación efectiva entre los equipos de negocio y de tecnología involucrados en el proceso de ETL. Estos aspectos garantizaron una comprensión clara de los requisitos, la correcta implementación de las transformaciones de datos y la detección temprana de posibles problemas.

En conclusión, este estudio proporciona una guía práctica y exhaustiva de mejores prácticas para optimizar los trabajos de ETL, destaca la importancia del procesamiento paralelo, la gestión de errores, la reutilización de recursos y la colaboración entre equipos. La implementación de estas soluciones contribuyó a mejorar la eficiencia, la calidad y el rendimiento de los procesos de ETL, lo que garantizó una correcta transformación y carga de datos en entornos empresariales (Seenivasan, ETL (Extract, Transform, Load) Best Practices , 2023).

Mejora del rendimiento de los trabajos ETL

Este estudio se enfoca en abordar el problema de optimizar los trabajos de ETL, centrándose en técnicas de paralelismo y soluciones de optimización. Se discuten varios aspectos clave, como la extracción eficiente de datos, la reducción de redundancia, el procesamiento en paralelo, la partición de datos, el almacenamiento en caché, la carga incremental, el monitoreo y la evaluación del rendimiento.

El documento propone soluciones para mejorar el rendimiento y la eficiencia de los trabajos de ETL. Entre las soluciones mencionadas se encuentran el monitoreo regular del rendimiento, la optimización de los procesos de extracción de datos, el uso de herramientas de monitoreo y métricas de rendimiento, la aplicación de técnicas de paralelismo, y la implementación de prácticas efectivas de gobierno y gestión de datos.

Se destaca la importancia del monitoreo continuo del rendimiento de los trabajos de ETL para identificar posibles problemas y áreas de mejora. Además, se mencionan técnicas de paralelismo que se pueden aplicar, como el paralelismo basado en procesos, el paralelismo basado en clúster, el paralelismo basado en tuberías y el procesamiento paralelo basado en la nube. Cada técnica se adapta a diferentes escenarios y requisitos, y se enfatiza la importancia de seleccionar la estrategia de paralelismo adecuada según las características de los datos y los recursos disponibles.

En conclusión, este estudio propone soluciones y enfoques para mejorar el rendimiento de los trabajos de ETL, con el uso de técnicas de paralelismo y optimización. Al implementar estas soluciones, las organizaciones pueden lograr un pipeline de datos más eficiente y efectivo, lo que reduce la redundancia, optimiza los procesos y garantiza una gestión adecuada de los datos. El monitoreo continuo y la selección adecuada de técnicas de paralelismo son aspectos fundamentales para mejorar el rendimiento general de los trabajos de ETL. (Seenivasan, Improving the Performance of the ETL Jobs, 2023)

Una vez revisados detenidamente los estudios relacionados, se emprende el siguiente paso en la investigación, en el que se aborda los conceptos generales en el próximo capítulo. Este enfoque progresivo permitirá una comprensión más completa y detallada de la materia, que contribuye al avance significativo de los objetivos planteados en la investigación.

Capítulo II

Este capítulo, presenta la exploración exhaustiva de los conceptos fundamentales que sustentan el presente estudio. Se abordan de manera detallada elementos esenciales como la optimización, la concurrencia, el paralelismo y el uso de hilos. Se define su importancia y contribución al marco conceptual que sustenta nuestra propuesta. Además, se profundiza en aspectos clave como la normativa ISO 25010, que proporcionan una base sólida para evaluar y comparar las características de nuestro enfoque. La atención se extiende también hacia los motores de texto de las bases de datos, que referencian su papel crucial en la eficiencia y efectividad de nuestros sistemas. A lo largo de este análisis, se exploran conexiones fundamentales con herramientas y tecnologías específicas, se destaca la relevancia de Python y otras plataformas afines que enriquecen nuestra comprensión y aplicabilidad en el ámbito de estudio propuesto. En conjunto, este capítulo actúa como un cimiento integral que ilumina las complejidades y sinergias inherentes a la propuesta que se desarrolla en el presente trabajo.

Marco teórico

Para el presente marco teórico, se observa el ámbito de la optimización del proceso de extraer, transformar y cargar (ETL) en entornos de Big Data, centrándonos específicamente en el contexto empresarial de "Casa de Incentivos Casintour" durante el año 2023. El proceso ETL despliega un papel crítico en la gestión y análisis de grandes volúmenes de datos, permite la extracción eficiente de información desde diversas fuentes, su transformación de acuerdo con las necesidades específicas y finalmente, su carga en un almacén de datos destinado a facilitar la toma de decisiones informadas.

La empresa "Casa de Incentivos Casintour" se enfrenta a los desafíos inherentes a la manipulación de grandes cantidades de datos, que requiere una revisión integral de su proceso ETL para maximizar la eficiencia y minimizar los tiempos de procesamiento. En este contexto, es crucial explorar y comprender las técnicas de paralelismo basado en procesos, una estrategia avanzada que busca distribuir la carga de trabajo a través de múltiples procesadores o nodos, con el fin de mejorar la velocidad y el rendimiento general del proceso ETL en un entorno de Big Data.

Para contextualizar adecuadamente este estudio, se abordan los fundamentos teóricos de Big Data, se destaca su impacto en la empresa moderna y la evolución de los métodos de procesamiento de datos. Se exploran los principios esenciales del proceso ETL, se analizan los elementos clave de extracción, transformación y carga, así como sus desafíos particulares en el contexto de grandes volúmenes de datos.

Además, se examina, en profundidad las técnicas de paralelismo, se evalúa cómo la distribución eficiente de tareas puede redundar en mejoras significativas en la velocidad de procesamiento y, por ende, en la capacidad de respuesta del sistema. Esto implica una revisión

detallada de conceptos como el paralelismo de datos, el paralelismo de tarea y sus aplicaciones específicas en el ámbito del proceso ETL.

A lo largo de este marco teórico, también se abordarán las herramientas y tecnologías relevantes en el dominio de Big Data y ETL, que demostró ser eficaces en entornos empresariales similares. Este análisis permitirá fundamentar la elección de enfoques específicos para la optimización del proceso ETL en "Casa de Incentivos Casintour", lo que proporciona un marco teórico sólido y orientado a la aplicación práctica de estrategias de paralelismo basado en procesos durante el año 2023.

Optimización

Es así que, para iniciar, se tomó la definición de optimización expuesta por IBM que afirma que la optimización en programación hace referencia al proceso de modificar el software para que algunos aspectos del mismo funcionen de manera más eficiente y/o utilicen menos recursos, lo que resulta en un mejor rendimiento. La optimización puede centrarse en mejorar uno o dos aspectos del rendimiento, como el tiempo de ejecución, el uso de la memoria, el espacio en disco, el rendimiento y otros (IBM, 2023), misma definición que permite comprender la importancia de optimizar un programa. Como en nuestro caso, se ha visto la necesidad de reducir el tiempo de ejecución y aprovechar los recursos que se tienen a disposición.

Características de la optimización de software. Las características de una buena optimización del software incluyen:

- Mejora del rendimiento: La optimización busca mejorar la velocidad de ejecución, el uso de memoria, el espacio en disco, el ancho de banda, entre otros aspectos del rendimiento.
- Niveles de optimización: La optimización puede darse en varios niveles, como la optimización en la fase de diseño, la optimización del código fuente de la aplicación, la optimización en el ámbito del montaje, entre otros.

- Código de buena calidad: Un código de buena calidad es fundamental en la optimización de software.
- Eficiencia en el uso de recursos: La optimización permite que los recursos de hardware se utilicen de forma más eficiente.
- Automatización y herramientas: La optimización puede ser automatizada por compiladores o realizada con el uso de diversas herramientas.

Estas características son fundamentales para lograr una optimización efectiva del software, lo que a su vez contribuye a un mejor desempeño y eficiencia del sistema (Luengo, 2023). En resumen, la optimización de software es un proceso continuo que implica la búsqueda constante de mejoras en el rendimiento y la eficiencia, a través de la aplicación de buenas prácticas y la utilización de herramientas avanzadas de desarrollo. Este enfoque integral es esencial para mantener sistemas informáticos ágiles y competitivos en un entorno tecnológico en constante evolución.

Optimización en Función del Tiempo. La reducción del tiempo para optimizar el software es importante porque permite mejorar la eficiencia en el proceso de desarrollo y entrega de productos, lo que a su vez puede reducir costos y mejorar la satisfacción del cliente. Además, la optimización del software puede mejorar el rendimiento de la aplicación o programa, lo que puede ser crucial en entornos donde el tiempo de respuesta es crítico. Al reducir el tiempo necesario para optimizar el software, se pueden obtener resultados más rápidos y eficientes, lo que puede ser beneficioso tanto para la empresa como para los usuarios finales (De DocuSign, C, 2021).

Normativa ISO

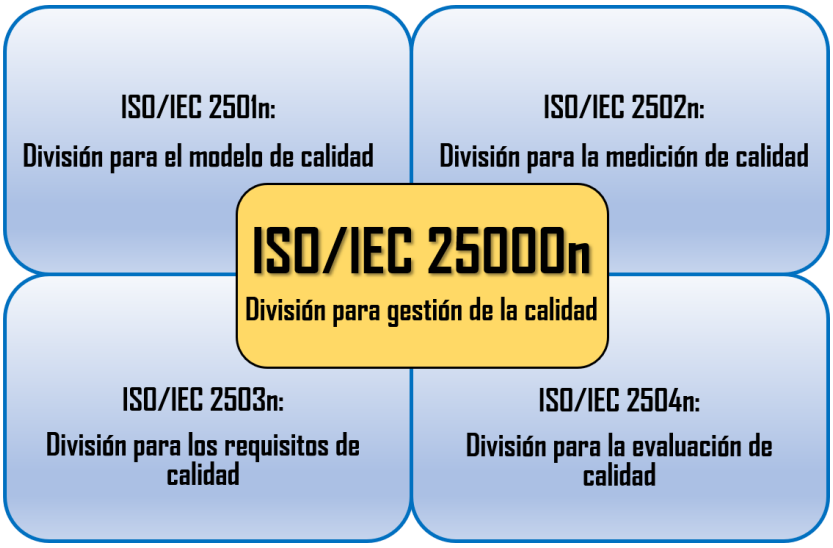
Para reconocer que hubo una mejora en el programa, se debe tener un punto de partida. Por ello, se ha escogido un estándar que permite medir en base a varias métricas el

estado actual del programa y su estado final luego de aplicar un proceso de optimización con el uso de técnicas de programación que se serán explicadas más adelante. En cuanto al estándar, se escogió la Norma ISO 25010.

ISO 25000. Para tener una idea clara de esta norma se debe primero definir la ISO 25000, la cual comprende reglas internacionales para la calidad de sistemas y software. Existe una división de calidad que es una de las cinco divisiones principales y una división de extensión. Existen cuatro: "calidad de la data", "calidad del servicio", "calidad del producto" y "calidad de uso" (Perdomo & Zapata, 2021). En esta serie, dos de los cuatro modelos de calidad, a saber, modelo de calidad del producto y el modelo de calidad de uso, se describen en la norma ISO/IEC 25010:2011, como se muestra en la Figura 2.

Figura 1

División de la ISO 25000



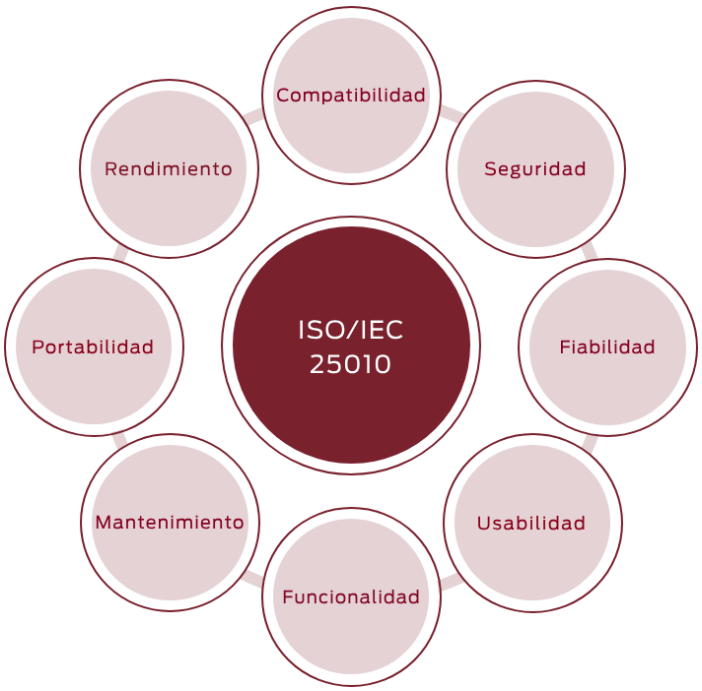
Nota. La imagen muestra la división de gestión de la calidad de la norma ISO 25000. Tomada de ISO/IEC 25000 - CALIDAD DE SOFTWARE. (s. f.-d). https://iso25000quality.blogspot.com/

ISO 25010. La ISO 25010 proporciona un marco de referencia para poder evaluar la calidad del producto de software, define un conjunto de características y subcaracterísticas que pueden ser utilizadas como directrices para la evaluación.

Por ejemplo, se han definido 81 métricas para usabilidad, fiabilidad, mantenibilidad, seguridad y evolucionabilidad con el uso del enfoque de Goal-Question-Metric (Schramme & Macías, 2019). También se han definido métricas para características como adecuación funcional, eficiencia de rendimiento, compatibilidad y portabilidad, que pueden observarse en la Figura 3 más a detalle.

Figura 2

Normativa ISO 25010



Nota. La imagen muestra las métricas que componen las ISO 25010.Tomado de Normas de Calidad < XITASO (2022) XITASO. Disponible en: https://xitaso.com/es/compania/normas-de-calidad/.

El estándar aborda aspectos clave que impactan directamente en la satisfacción del usuario y el rendimiento eficiente del software en una variedad de aplicaciones y entornos. Está

ha evolucionado para abordar los desafíos cambiantes en el desarrollo de software, que proporciona un marco completo, como se observa en la Figura 1. Su adopción ha crecido significativamente, convirtiéndose en un estándar reconocido y respetado en la industria del desarrollo de software.

Figura 3

Organigrama de la norma ISO 25010

Nota. La decisión de adoptar un enfoque específico queda a discreción y conveniencia de aquellos responsables de la producción y mantenimiento del producto. Tomada de ISO 25000 por (ISO 25010. (s. f.-b). https://iso25000.com/index.php/normas-iso-25000/iso-25010

En el contexto de la ISO 25010, la eficiencia de desempeño se refiere a la capacidad del software para utilizar los recursos de manera óptima, evitar la generación de cuellos de botella y asegurar una ejecución eficiente de las tareas asignadas. Se centra en aspectos como la rapidez en la respuesta a las solicitudes, el uso eficiente de la memoria y la minimización del consumo de recursos (Desempeño, n.d.), como se explica más a detalle en el siguiente apartado.

Métrica de Eficiencia de Desempeño de la Normativa ISO 25010. La norma ISO/IEC 25010 define la métrica de eficiencia de desempeño como una subcaracterística de calidad que evalúa el comportamiento temporal y el uso de recursos de un software. Esta métrica se centra en aspectos como el tiempo de respuesta, la utilización de recursos y la capacidad del software para realizar sus funciones de manera eficiente. La eficiencia de desempeño es crucial para garantizar que el software funcione de manera rápida y efectiva, especialmente en entornos donde se requiere un alto rendimiento.

Por ejemplo, la métrica de eficiencia de desempeño puede incluir la evaluación del tiempo de respuesta del software en diferentes situaciones, la medición de la utilización de recursos como la memoria y el procesador, y la capacidad del software para manejar grandes volúmenes de datos de manera eficiente.

En resumen, la métrica de eficiencia de desempeño definida en la norma ISO/IEC 25010 es fundamental para evaluar la capacidad de un software para funcionar de manera eficiente en términos de tiempo y recursos (Falco & Robiolo, 2021).

Técnicas para la Optimización

Asimismo, no hay que restarle importancia a la técnica que se elige para cumplir con el objetivo. Por consiguiente, se verá las definiciones de dos técnicas que se han tomado como punto de partida, el paralelismo y concurrencia; como menciona (Fahimeh , Jie, & Farookh, 2013), la utilización del procesamiento paralelo y los sistemas informáticos distribuidos también se ha identificado como una estrategia clave para mejorar el rendimiento de ETL para poder paralelizar problemas inicialmente secuenciales que pueden ser resueltos al ser programados con técnicas de concurrencia.

Concurrencia. Por un lado, se tiene la concurrencia que, en programación, se refiere a la capacidad de distintas partes de un programa, algoritmo o problema de ser ejecutadas en desorden o en orden parcial, sin afectar el resultado final. Los cálculos pueden ser ejecutados en múltiples procesadores o en distintos hilos de ejecución. La concurrencia permite diseñar software que ejecute eventos o situaciones que suceden o existen al mismo tiempo (Tu, Liu, Song, & Zhang, 2019).

Algunas de sus características son:

- Gestión de procesos independientes que requiere coordinación de recursos compartidos.
- Existen dos modelos de programación concurrente: memoria compartida y paso de mensajes.
- Los módulos concurrentes son de dos tipos: proceso (instancia de un programa en ejecución aislado de otros procesos) e hilo o thread (lugar de control dentro de un programa en ejecución).
- La concurrencia es esencial en la programación moderna, por ejemplo, en sitios web, aplicaciones móviles e interfaces gráficas de usuario.

Paralelismo. Por otro lado, el "paralelismo" se refiere a la técnica de dividir problemas de gran magnitud en partes más pequeñas que se resuelven simultáneamente, ya sea mediante la ejecución de múltiples instrucciones al mismo tiempo o mediante la utilización de varios procesadores o núcleos de procesamiento. Este enfoque permite aprovechar al máximo los recursos computacionales disponibles para resolver problemas de manera más eficiente. El paralelismo puede ser implementado a través de lenguajes, API, frameworks y otras herramientas que permiten aprovechar el potencial del hardware de forma paralela (Yan, et al., 2020).

En este caso se elige al paralelismo como técnica para optimización de Maricruz, ya el mismo es esencial para mejorar el rendimiento y la eficiencia de los sistemas, especialmente en entornos donde se requiere procesar grandes cantidades de datos o realizar múltiples tareas simultáneamente.

Comparativa entre Paralelismo y Concurrencia. Es importante diferenciar el paralelismo de la concurrencia, ya que mientras el paralelismo se enfoca en la ejecución simultánea de tareas, la concurrencia se relaciona más con el diseño del software y puede llevarse a cabo incluso con una sola unidad de procesamiento, en la Tabla 1 se puede observar una comparativa más a detalle de estos dos conceptos.

Tabla 1

Comparativa de características del paralelismo y concurrencia

Parámetro	Concurrencia	Paralelismo
Ejecución de tareas	Tareas independientes se ejecutan de manera concurrente, pero no necesariamente al mismo tiempo.	Tareas se ejecutan simultáneamente, se divide la carga de trabajo entre varios procesadores o núcleos.

Parámetro	Concurrencia	Paralelismo
Interferencia	Puede haber interferencia entre tareas concurrentes debido a la compartición de recursos.	Menos interferencia ya que las tareas se ejecutan de manera independiente en distintos hilos o procesadores.
Coordinación	Requiere coordinación explícita para sincronizar el acceso a recursos compartidos.	Menor necesidad de coordinación, ya que las tareas no comparten recursos de manera directa.
Complejidad	Más complejo gestionar la concurrencia debido a la posibilidad de condiciones de carrera y bloqueos.	Menos complejidad en la gestión de tareas paralelas, pero aún puede haber desafíos de sincronización.
Comportamiento con hilos	Puede ejecutarse en un solo hilo o en múltiples hilos. La ejecución en múltiples hilos puede no aprovechar completamente los recursos.	Aprovecha mejor los recursos al ejecutarse en múltiples hilos o procesadores y se distribuye la carga de trabajo de manera eficiente.

Nota. Se muestra una comparative en base a determinadas características para diferenciar entre paralelismo y concurrencia. Basado en Kumar, S. (2023). Introduction to Parallel Programming. Cambridge University Press.

Big Data

Una vez que se ha revisado las dos importantes técnicas computacionales, aparece un concepto importante como lo es Big Data, que se refiere a conjuntos masivos de datos los cuales son extremadamente grandes, variados y complejos, lo que plantea desafíos en términos de almacenamiento, análisis y visualización. Estos conjuntos de datos provienen de una amplia variedad de fuentes, como transacciones en línea, correos electrónicos, videos,

audios, imágenes, registros, publicaciones en redes sociales, datos científicos, sensores y aplicaciones móviles.

De acuerdo a (Bourany, 2018), en su libro menciona que para que una empresa pueda distinguir si su información es Big Data debe basarse en las 5v que se describen a continuación:

- Volumen: la gran cantidad de información contenida en estas bases de datos.
- Velocidad: la velocidad de su creación, recopilación, transmisión y análisis.
- Variedad: diferencias de naturaleza, formatos y estructuras.
- Valor: la capacidad de estos datos para generar beneficios.
- Veracidad: su validez, es decir, calidad y precisión, así como su fiabilidad.

El análisis de Big Data puede proporcionar información valiosa para la toma de decisiones, identificación de patrones y tendencias, y descubrimiento de conocimientos significativos. Como en el caso de la empresa Casa de Incentivos Caintour Sin embargo, también plantea desafíos en términos de privacidad, seguridad y la necesidad de infraestructuras y herramientas especializadas (Sagiroglu & Sinanc, 2013).

Almacenamiento de Big Data. Como se menciona, el almacenamiento de Big Data a menudo requiere tecnologías y métodos específicos, como bases de datos NoSQL, sistemas de archivos distribuidos y soluciones de almacenamiento en la nube, pero en este caso, y dado que la empresa Casa de Incentivos Casintour ya maneja cierto tipo de tecnologías, se ha exigido trabajar con ellas. Otro concepto importante relacionado con el almacenamiento son los motores de almacenamiento: MyISAM e InnoDB, los cuales se revisarán a continuación.

Motores de Almacenamiento

MyISAM. Se empieza por MyISAM, es un motor de almacenamiento en MariaDB no transaccional y puede optimizar la base de datos con un modelo no relacional. Admite tres prioridades de acceso, que incluyen: LOW_PRIORITY, DELAYED y HIGH_PRIORITY, y se utiliza para tablas que no requieren relaciones entre los datos. Además, MyISAM determina la prioridad de los comandos de Lenguaje de Manipulación de Datos (DML), como LOW_PRIORITY, DELAYED y HIGH_PRIORITY.

InnoDB. En cuanto a InnoDB, en MariaDB es un motor de almacenamiento que soporta relaciones entre tablas y se conoce como base de datos transaccional. Está diseñado para garantizar la integridad de los datos y el cumplimiento de ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). InnoDB es ampliamente utilizada por su fiabilidad, control de concurrencia y capacidad de recuperación ante fallos (Sucipto, Resti, Andriyanto, Karaman, & Qamaria, 2019).

Elección del motor de almacenamiento. La elección de usar el motor de almacenamiento MyISAM en lugar de InnoDB (u otros motores de almacenamiento) en MariaDB puede depender de varios factores, y es importante tener en cuenta las características y limitaciones de cada motor de almacenamiento.

MyISAM ha sido históricamente preferido en ciertos escenarios debido a sus características específicas. Algunas de las razones positivas para elegir MyISAM en situaciones que implican paralelismo basado en procesos podrían ser:

- Bloqueo de tabla: MyISAM utiliza un enfoque de bloqueo de tabla en lugar de un bloqueo de fila como InnoDB. Esto significa que las consultas de lectura no bloquearán otras consultas de lectura. En situaciones donde se realizan principalmente operaciones de lectura y se necesita un mejor rendimiento de lectura simultánea, este enfoque puede ser beneficioso.
- Velocidad de lectura: MyISAM tiende a ser más rápido en operaciones de lectura simples en comparación con InnoDB. Si su aplicación implica principalmente lecturas y las actualizaciones o escrituras son menos frecuentes, MyISAM podría ofrecer un mejor rendimiento en términos de velocidad de lectura.
- Uso eficiente del almacenamiento: MyISAM puede ser más eficiente en términos de espacio en disco para ciertos tipos de tablas y datos. Por ejemplo, en tablas que contienen principalmente datos estáticos y no necesitan soportar transacciones, MyISAM podría ser una opción viable.

Sin embargo, es importante tener en cuenta que MyISAM tiene limitaciones significativas en comparación con InnoDB. Algunas de las desventajas incluyen la falta de soporte para transacciones ACID, la ausencia de integridad referencial, y el bloqueo de tabla que puede afectar negativamente la concurrencia en entornos con muchas escrituras simultáneas.

En entornos modernos, especialmente en aplicaciones que requieren transacciones y concurrencia, InnoDB es a menudo la elección preferida debido a su soporte para características como transacciones, bloqueo de fila, y recuperación ante fallas. Siempre es

recomendable evaluar las necesidades de rendimiento antes de decidir qué motor de almacenamiento utilizar (MariaDB, n.d.). En la Tabla 2 se detallan las características de cada motor de almacenamiento en base a determinados parámetros.

Tabla 2

Comparativa entre motores de almacenamiento

Parámetro	InnoDB	MyISAM
Motor de almacenamiento	Transaccional y orientado a transacciones ACID.	No es transaccional y no ofrece soporte completo ACID.
Bloqueo de nivel de fila	Utiliza bloqueo de nivel de fila, lo que facilita la concurrencia en lecturas.	Utiliza bloqueo de nivel de tabla, lo que puede generar conflictos en entornos concurrentes.
Rendimiento en lecturas	Buen rendimiento en lecturas gracias al bloqueo a nivel de fila y caché de índices.	Rendimiento rápido en lecturas debido al bloqueo a nivel de tabla y a la ausencia de overhead transaccional.
Recuperación después de fallos	Ofrece una recuperación más robusta después de fallos gracias a la recuperación de transacciones.	Menos robusto en la recuperación después de fallos, puede requerir reparación manual de tablas.
Integridad referencial	Admite claves foráneas y garantiza la integridad referencial.	No admite claves foráneas y no garantiza la integridad referencial.
Comportamiento con hilos	InnoDB puede experimentar cierta complejidad en entornos de escritura intensiva debido a su bloqueo a nivel de fila.	MyISAM puede ser más directo en el manejo de múltiples peticiones, especialmente en escenarios

Parámetro	InnoDB	MyISAM
		con muchas solicitudes concurrentes.

Nota. La tabla muestra una comparativa entre dos moteres de almacenamiento. Datos tomados de Petrov, P., Kuyumdzhiev, I., Dimitrov, G., & Kremenska, A. (2022). Relative Performance of Various Types of Repositories for MySQL Archive Backup and Restore Operations. International Journal of Online & Biomedical Engineering, 18(13).

Lenguaje de programación: Python

Python es un lenguaje de programación de propósito general que se destaca por su simplicidad y legibilidad. Es ampliamente utilizado en diversos campos, que incluyen el análisis de datos, desarrollo web, inteligencia artificial, entre otros, gracias a su amplia gama de bibliotecas y su enfoque en la productividad del programador.

Python ofrece varias bibliotecas que permiten el paralelismo basado en procesos, lo que permite ejecutar tareas en paralelo con el uso de múltiples procesos en lugar de hilos. Estas bibliotecas facilitan la creación y administración de procesos paralelos, que proporcionan una interfaz simple y eficiente (Dalcin, Paz, Kle, & Cosimo, 2011).

Algunas de las bibliotecas populares de paralelismo basado en procesos en Python son:

- Multiprocessing: Esta biblioteca proporciona una forma sencilla de crear y administrar procesos en Python. Permite la ejecución de tareas en paralelo con el uso de procesos individuales y ofrece mecanismos para compartir datos y comunicarse entre los procesos.
- Concurrent.futures: Esta biblioteca es parte de la biblioteca estándar de Python y proporciona una interfaz de alto nivel para la ejecución de tareas en paralelo. Ofrece abstracciones para la ejecución concurrente de código con el uso de hilos o procesos,

que incluyen la clase ProcessPoolExecutor que permite ejecutar tareas en paralelo que usan un grupo de procesos.

- Estas bibliotecas y otras herramientas relacionadas con el paralelismo basado en procesos en Python facilitan la implementación de tareas concurrentes y paralelas, lo que puede mejorar significativamente el rendimiento y la eficiencia de los programas.
- Clase ProcessPoolExecutor de Python: es una clase de la biblioteca concurrent.futures en Python que proporciona una interfaz para ejecutar tareas de manera paralela con el uso de múltiples procesos. Permite aprovechar el paralelismo en sistemas multiproceso para acelerar la ejecución de tareas.

Modelos matemáticos de la librería ProcessPoolExecutor de Python. Los modelos matemáticos de la librería ProcessPoolExecutor en Python se refieren a los algoritmos y enfoques matemáticos utilizados para implementar el paralelismo basado en procesos en esta librería. Estos modelos pueden incluir cálculos de eficiencia, escalabilidad y rendimiento, así como técnicas de asignación y distribución de tareas en los procesos.

Librería time de Python: Proporciona funciones relacionadas con la medición y manipulación del tiempo en un programa. Permite medir el tiempo de ejecución de una porción de código, crear retardos o pausas en la ejecución del programa y realizar operaciones relacionadas con la fecha y hora (Sodian, Wen, Davidson, & Loskot, 2022).

Patrón de diseño Factory

El patrón de método de fábrica es una técnica de diseño que se basa en la definición de una clase abstracta o interfaz para la creación de objetos, a la que se delega la responsabilidad de instanciar la clase a sus subclases o clases de implementación. Este enfoque permite que una clase principal difiere la creación de instancias a sus subclases, lo que brinda flexibilidad y extensibilidad al sistema.

Este enfoque modulariza el proceso de creación de objetos y permite la introducción de nuevas subclases sin modificar la clase principal. Además, facilita la gestión de la lógica de creación de instancias, ya que cada subclase puede ajustarla según sus propias reglas y requisitos (Yuhua, Miao , & Limei, 2011).

El patrón de diseño Factory ofrece varios beneficios, como ya se mencionó antes, se incluye la creación de nuevos objetos a través de una interfaz común sin exponer los detalles sobre el tipo de objetos. Esto permite una mayor flexibilidad y extensibilidad en el código, ya que se pueden agregar nuevos tipos de objetos sin modificar el código existente. Además, el patrón Factory promueve el principio de encapsulamiento al ocultar la lógica de creación de objetos, lo que facilita el mantenimiento y la reutilización del código. Esto también mejora la legibilidad del código al separar la lógica de creación de objetos de la lógica de negocio principal.

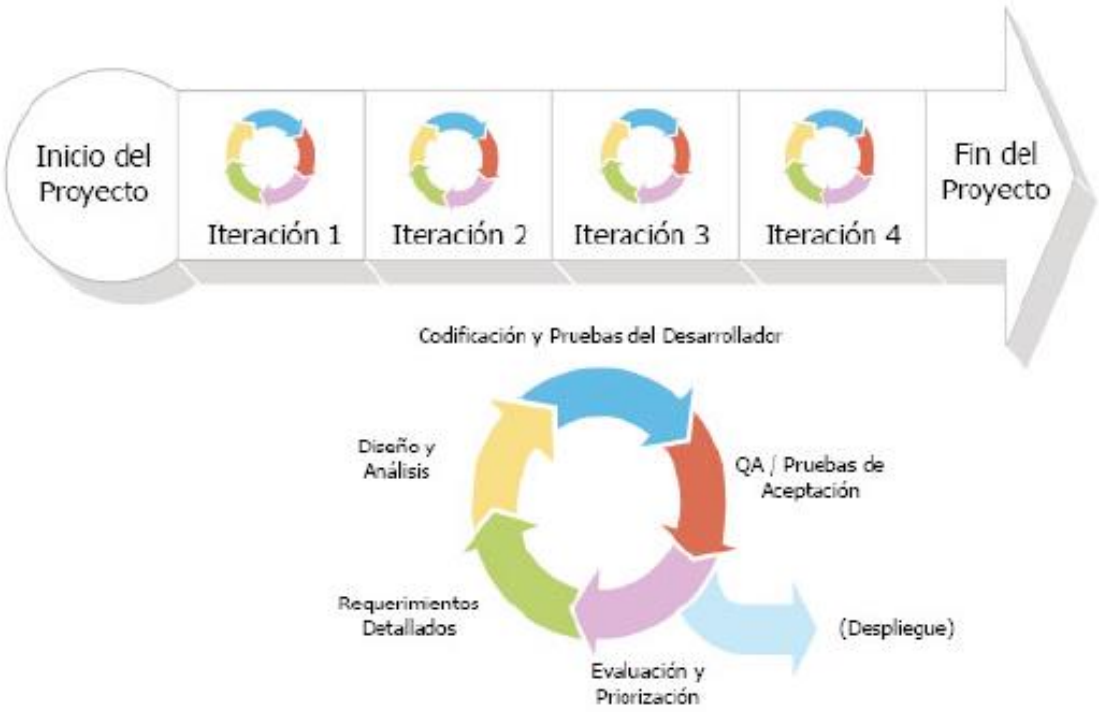
En resumen, el patrón de diseño Factory proporciona flexibilidad, extensibilidad, encapsulamiento, centralización y legibilidad al facilitar la creación de objetos a través de una interfaz común (Abukwaik, Gogolev, Groß, & Aleksy, 2020).

Metodología Ágil de Desarrollo

Ágil es un término utilizado para identificar: “una metodología de desarrollo de software que se caracteriza por la capacidad de adaptarse rápidamente y eficientemente a cambios en los requisitos y en las condiciones del mercado” (Visual Paradigm, n.d.). Un ejemplo de ello es la metodología Scrum, cuyo proceso de iteración se visualiza en la Figura 4.

Figura 4

Proceso de Iteración



Nota. Se muestra el proceso de iteración que se aplica en la metodología Scrum. lustración tomada de Bernal, J. (2023, November 23). Scrum – Gestión Ágil de Proyectos I. BLMovil. https://www.blmovil.com/scrum-gestion-agil-de-proyectos-i/

Eventos Scrum. Los eventos de Scrum, que comprenden la planificación del sprint, el Scrum diario, la revisión del sprint y la retrospectiva del sprint, se encuentran meticulosamente estructurados dentro del marco de trabajo de Scrum. La observancia rigurosa de estos eventos resulta crucial para el éxito en la implementación de Scrum, ya que contribuye significativamente a la percepción de un equipo como exitoso en el contexto de Scrum. La realización de estos eventos en un mismo tiempo y lugar, con límites temporales bien definidos, no solo establece regularidad, sino que también minimiza la necesidad de convocar reuniones adicionales fuera de los eventos de Scrum, conforme lo recomienda la guía de Scrum (Kadenic, Koumaditis, & Junker-Jensen, 2023).

Es fundamental subrayar que la guía de Scrum enfatiza la importancia de llevar a cabo todos los eventos de Scrum de acuerdo al marco de trabajo establecido, y cualquier desviación de este protocolo podría comprometer la esencia misma de Scrum. Esta adhesión estricta a las prácticas delineadas en la guía asegura una implementación coherente y exitosa del marco de trabajo de Scrum.

Por otro lado, el papel del Scrum Master emerge como esencial en el mantenimiento de la integridad de los eventos de Scrum. Su capacidad para garantizar la realización puntual de estos eventos dentro de los límites de tiempo establecidos se erige como un factor crítico para el éxito en la implementación de Scrum. La figura del Scrum Master, como facilitador y protector del proceso Scrum, desempeña un papel central en la cohesión y eficacia del equipo.

En síntesis, los eventos de Scrum no solo constituyen pilares fundamentales para la implementación exitosa de Scrum, sino que su estructura y ejecución de acuerdo al marco de trabajo de Scrum son determinantes para el progreso y éxito continuo del equipo en el entorno Scrum.

Los eventos de Scrum incluyen:

- Planificación del sprint: En esta reunión, el equipo de Scrum define el objetivo del sprint y crea un plan para entregar el trabajo necesario para alcanzar ese objetivo. Esta reunión está estructurada de acuerdo al marco de trabajo de Scrum (Kadenic, Koumaditis, & Junker-Jensen, 2023).
- Scrum diario: Esta es una reunión diaria corta en la que el equipo de Scrum se reúne para compartir actualizaciones sobre el progreso del trabajo, identificar cualquier impedimento y planificar el trabajo para el próximo día. Esta reunión también está estructurada de acuerdo al marco de trabajo de Scrum (Kadenic, Koumaditis, & Junker-Jensen, 2023).
- Revisión del sprint: Al final de cada sprint, el equipo de Scrum presenta el trabajo completado al cliente o a los stakeholders, recibe retroalimentación y ajusta el backlog del producto según sea necesario. Esta reunión también está estructurada de acuerdo al marco de trabajo de Scrum (Kadenic, Koumaditis, & Junker-Jensen, 2023).
- Retrospectiva del sprint: Después de la revisión del sprint, el equipo de Scrum se reúne para reflexionar sobre el sprint, identificar qué salió bien, qué salió mal y cómo pueden mejorar en el futuro. Esta reunión también está estructurada de acuerdo al marco de trabajo de Scrum (Kadenic, Koumaditis, & Junker-Jensen, 2023).

Estos eventos son fundamentales para la implementación exitosa de Scrum y deben realizarse de acuerdo al marco de trabajo de Scrum para garantizar la regularidad, consistencia y predictibilidad en el proceso de desarrollo.

Además de los eventos, los artefactos en Scrum desempeñan un papel crucial en el éxito y la eficacia del marco de trabajo. Estos artefactos, que incluyen la lista de producto, la lista de sprint y el incremento, actúan como herramientas esenciales para la transparencia y la gestión efectiva del trabajo en el contexto de Scrum. La lista de producto, por ejemplo, sirve

como una representación detallada de los requisitos del proyecto, mientras que la lista de sprint proporciona una visión clara de las tareas que deben abordarse durante un sprint específico. El incremento, por su parte, representa la suma de todos los elementos completados durante un sprint, que refleja así el progreso tangible del equipo.

La coherente utilización y actualización de estos artefactos según las directrices de Scrum no solo fomentan la transparencia, sino que también facilitan la toma de decisiones informadas y la adaptación ágil a medida que avanza el proyecto. En última instancia, la comprensión y aplicación adecuada de estos artefactos fortalece la estructura y la eficiencia del proceso Scrum, lo que contribuye de manera significativa al logro de los objetivos del equipo y del proyecto.

Artefactos del Scrum.

- User stories: Breve descripción de un deseo desde el punto de vista del usuario final, que ayuda a entregar elementos de trabajo completamente realizados en cada iteración (Baijens, Helms, & Iren, 2020).
- Product backlog: Lista completa de deseos de las partes interesadas con respecto al producto, que proporciona una visión general de en qué puede trabajar el equipo en futuras iteraciones (Baijens, Helms, & Iren, 2020).
- Sprint backlog: Lista de elementos a desarrollar durante un sprint, creada durante el refinamiento basado en los elementos del product backlog (Baijens, Helms, & Iren, 2020).
- Increment: Entregable de un sprint que consiste en varias user stories que juntas resultan en un producto de trabajo o semiacabado (Baijens, Helms, & Iren, 2020).

En la Figura 5 se muestran los artefactos más importantes de la metodología ágil Scrum, cuyos conceptos ya han sido definidos anteriormente. En última instancia, la correcta gestión de estos artefactos es esencial para el éxito de un proyecto Scrum, ya que aseguran una comunicación clara, una planificación efectiva y entregas incrementales que se alinean con las expectativas de los stakeholders. Este enfoque estructurado facilita la adaptabilidad y la entrega continua de valor en el desarrollo de software.

Figura 5

Artefactos de metodología SCRUM

Nota. Demuestra los artefactos que hacen parte de la metodología ágil SCRUM. Recuperada de Chito, R. (n.d.). Artefactos de Scrum – BLOG – Planeta Buhoos – Una Comunidad 100% Virtual. https://blog.buhoos.com/artefactos-de-scrum/

Beneficios de Scrum. La integración de la metodología Scrum conlleva beneficios significativos para el desarrollo de software. Entre estos beneficios se destaca una mejora sustancial en la calidad del código, lo cual facilita el mantenimiento del sistema. Asimismo, se observa una optimización en la entrega del producto final, lo que contribuye a una mayor eficiencia en los procesos de desarrollo.

La adopción de Scrum también promueve una mejor colaboración y comunicación entre los miembros del equipo, lo que puede traducirse en una sinergia más efectiva y una respuesta más ágil a los cambios en las necesidades del proyecto. Este enfoque promueve la transparencia en el proceso, lo que le permite comprender mejor el progreso y tomar decisiones más informadas. A pesar de estos beneficios, es importante señalar que también se destacan desafíos en la implementación de Scrum. Entre estos desafíos se encuentra la necesidad de adquirir conocimientos y habilidades específicas, dificultades organizativas y la resistencia al cambio. Sin embargo, a pesar de estos obstáculos, la aplicación de Scrum puede ser una estrategia prometedora para potenciar el desarrollo de software en aquellas organizaciones que optan por este enfoque metodológico (Mahmood, Usmani, Ali, & Farooqui, 2020).

Una vez que se han revisado y comprendido a fondo los conceptos fundamentales abordados en el presente capítulo, se procede con el desarrollo subsiguiente, donde se exploran puntos clave que constituyen la esencia de la propuesta, se definen los procesos y se redactan los pasos seguidos para conseguir el resultado propuesto.

Capítulo III

En este capítulo, se profundiza en el proceso de desarrollo e implementación de la propuesta planteada. En este contexto, se persigue el objetivo fundamental de evaluar la versión actual del programa mediante los criterios establecidos por la norma ISO 25010. Asimismo, se lleva a cabo una evaluación exhaustiva de la versión optimizada que surge como resultado de la aplicación de todas las mejoras propuestas.

Desarrollo

Para la elaboración del presente estudio, se comienza con la identificación de los requisitos específicos establecidos por la empresa según el Apéndice 2 y Apéndice 3, que incluyen la utilización de un lenguaje determinado, en este caso, Python. Además, se lleva a cabo un análisis exhaustivo del código existente proporcionado por la empresa. Este análisis se realiza con el propósito de evaluar el estado actual del código, con ayuda del marco de referencia de la norma ISO 25010. Este enfoque permite abordar de manera integral y sistemática la calidad del software, que considera aspectos como la funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, y portabilidad, entre otros.

En este contexto, es importante destacar que, dentro del marco normativo de la ISO 25010, se enfocará de manera específica en el apartado de "Eficiencia de Desempeño" y "Fiabilidad". Esta elección se fundamenta en la necesidad de medir y mejorar dos aspectos cruciales para el desarrollo del presente proyecto: el uso de recursos del procesador y el comportamiento temporal de los procesos. Se propone evaluar cuánto tiempo demora el proceso ETL (extraer, transformar, cargar), un componente vital en el flujo de trabajo, y, al mismo tiempo, cuantificar de manera precisa los recursos del procesador que consume.

Este enfoque estratégico en la eficiencia de desempeño y fiabilidad se alinean con el objetivo central de nuestra propuesta: implementar técnicas de paralelismo basado en procesos

para optimizar estos dos parámetros críticos. La utilización de técnicas de paralelismo no solo tiene el potencial de reducir significativamente los tiempos de ejecución de los procesos, sino que también puede aprovechar de manera más eficiente los recursos disponibles del procesador, lo cual contribuye así a una mejora integral en la eficiencia operativa del sistema. En el próximo apartado, se explora con mayor profundidad la metodología propuesta y cómo se espera que estas técnicas de paralelismo generen un impacto en la eficiencia del sistema, que se respalda así la consecución de los objetivos establecidos por la empresa.

Proceso de desarrollo e implementación de paralelismo basado en procesos en programa Maricus

Se ha implementado una clase llamada Hilo que usa el patrón de diseño Factory en Python, esta clase permite la creación de hilos con diferentes configuraciones y funcionalidades según lo antes mencionado. El constructor de la clase Hilo acepta varios parámetros, como el método a ejecutar, un iterador, un arreglo, argumentos adicionales y funciones con argumentos.

La función ejecutar_multiples_funciones utiliza un ThreadPoolExecutor para ejecutar múltiples funciones en hilos concurrentes. Se crea un diccionario de Futures con las funciones y sus argumentos correspondientes, se espera a que todas las funciones finalicen y se recopilan los resultados en un diccionario que asocia el nombre de la función con su resultado. Cualquier error durante la ejecución se maneja adecuadamente, como se muestra en la Figura 6.

Figura 6

Primera función de la clase Hilos

```
def ejecutar_multiples_funciones(self):
    results_dict = {}

    with concurrent.futures.ThreadPoolExecutor() as executor:
        try:
            futures = {executor.submit(func, *args): func if callable(func) else func[0] for func, args in self.funciones_con_args}
            concurrent.futures.wait(futures)

            for future, func_ref in futures.items():
                try:
                    result = future.result()
                    func_name = func_ref.__name__ if hasattr(func_ref, '__name__') else str(func_ref)
                    results_dict[func_name] = result
                except Exception as e:
                    print(f"Error en la ejecución de {func_name}: {traceback.format_exc()}")


        except Exception as e:
            print(traceback.format_exc() + "Error en la ejecución de funciones concurrentes")

    return results_dict
```

Nota. Función de la clase Hilos para la ejecución de múltiples funciones. Autoría Propia

La función generarHiloParametrosConRetorno utiliza un ProcessPoolExecutor para ejecutar un método en paralelo con los elementos de un arreglo y argumentos adicionales. Los resultados se almacenan en una lista y se manejan posibles excepciones, lo cual se evidencia en la Figura 7 que muestra la función.

Figura 7

Segunda función de la Clase Hilo

```
def generarHiloParametrosConRetorno(self):
    results = []
    n = psutil.cpu_count(logical=True)

    # Usar un solo executor para varias llamadas
    with concurrent.futures.ProcessPoolExecutor(max_workers=n) as executor:
        try:
            # Mapear las llamadas a la función método en paralelo
            results = list(executor.map(self.metodo, self.arreglo, [self.args]*len(self.arreglo)))

        except Exception as e:
            print(traceback.format_exc() + " Error en la ejecución del hilo")

    return results
```

Nota. Función de la clase Hilos para la ejecución de múltiples funciones. Autoría Propia

La clase HiloFactory actúa como una fábrica de hilos y proporciona dos métodos estáticos. crear_hilo_con_multiples_funciones crea un objeto Hilo con funciones y argumentos específicos y luego ejecuta esas funciones en hilos concurrentes. crear_hilo crea un objeto Hilo con parámetros personalizados, pero no ejecuta ninguna función.

Envío de parámetros a las funciones de la clase Hilo.

Parámetros para Función ejecutar_multiples_funciones.

consulta1 = pasos.consultas([(pasos.consultaNetviax, ()), (pasos.consultaBSP, ()),(pasos.consultaGrupoAerolineas,()), (pasos.consultarBoletosVoid, ())])

def consultas(funciones_con_args):
    resultados = hilo.crear_hilo_con_multiples_funciones(funciones_con_args)
    return resultados

En el código, se ejecutó una serie de consultas mediante la función `consultas`, la cual recibe un argumento llamado `funciones_con_args`. Este argumento es una lista que contiene tuplas, donde cada tupla tiene una función y sus argumentos asociados. En este caso, se utiliza

la clase `HiloFactory` para crear una instancia de la clase `Hilo` y ejecutar múltiples funciones concurrentemente.

La variable `consulta1` es una instancia de la lista de consultas. Cada elemento de la lista es una tupla que contiene una función de consulta (por ejemplo, `pasos.consultaNetviax`) y una tupla vacía de argumentos (en este caso, `()`). Estas tuplas se pasan como argumento a la función `consultas`.

Dentro de la función `consultas`, se utiliza la clase `HiloFactory` para crear un hilo con múltiples funciones (`hilo.crear_hilo_con_multiples_funciones(funciones_con_args)`). Esta función utiliza un `ThreadPoolExecutor` para ejecutar las funciones concurrentemente y devuelve los resultados recopilados en un diccionario.

Parámetros para Función generarHiloParametrosConRetorno.

```
    TicketsNuevos = pasos.Paso5(repoBoleto,args)
def Paso5(valores,args):
    try:
        iterador = []
        mi_hilo = hilo.crear_hilo(paso5,iterador,valores,args)
        resultados = mi_hilo.generarHiloParametrosConRetorno()


        for sublist in resultados:
            iterador.extend(sublist)

        return list(set(iterador))
    except:
        logger.error(traceback.format_exc())
```

En la segunda función, se envía parámetros a la función `Paso5`. La variable `args` contiene un diccionario con varios elementos, como "grupoaereo", "remarksCorp", "datoBoletos", y "ultimoPeriodo". Estos valores se utilizan como argumentos en la función `Paso5` junto con la lista `repoBoleto`.

Dentro de la función `Paso5`, se intenta crear un hilo con la clase `Hilo` mediante el método `crear_hilo`. Se pasa la función `paso5` como el método a ejecutar, una lista vacía como el iterador, la lista `valores` y el diccionario `args`. Luego, se utiliza el método `generarHiloParametrosConRetorno` de la instancia de la clase `Hilo` para ejecutar la función `paso5` en paralelo con los valores proporcionados y recopilar los resultados.

Los resultados se procesan para extender la lista `iterador` con sublistas obtenidas de la ejecución en paralelo. Finalmente, se retorna una lista única después de convertir `iterador` a un conjunto y luego de nuevo a una lista.

En caso de que se produzca una excepción durante la ejecución, se captura y se registra un mensaje de error con ayuda del módulo de registro y la función `traceback.format_exc()` para obtener detalles del error.

En resumen, esta estructura de código utiliza la clase `Hilo` para ejecutar la función `paso5` en paralelo con diferentes conjuntos de valores y argumentos, lo que facilita la tarea de procesamiento en lotes de datos y mejora la eficiencia del proceso.

La implementación del paralelismo basado en procesos en el código proporcionado es esencial para la optimización del proceso ETL en el contexto de Big Data, como se ilustra en el caso de la empresa "Casa de Incentivos Casintour". El uso de la clase `Hilo` y su método `generarHiloParametrosConRetorno` aprovecha el paralelismo de procesos mediante un `ProcessPoolExecutor`. Este enfoque permite ejecutar las tareas de extracción, transformación y carga (ETL) de manera eficiente al distribuir las operaciones en múltiples procesos, con lo cual se aprovecha así la capacidad de procesamiento paralelo de la máquina.

Dado que el proceso ETL en entornos de Big Data implica el manejo de grandes volúmenes de datos, la ejecución concurrente de tareas puede significativamente acelerar el rendimiento general del sistema. La capacidad de procesamiento en paralelo proporcionada por el código contribuye a una mayor velocidad en la ejecución de las operaciones, lo que permite reducir el tiempo total necesario para completar el proceso ETL. Esto resulta crucial para empresas que manejan grandes conjuntos de datos, ya que permite una mejora significativa en la eficiencia y la capacidad de respuesta del sistema, lo que optimiza así el flujo de trabajo ETL durante el año 2023 para "Casa de Incentivos Casintour".

Para evaluar y comprobar la presencia de una optimización en el programa de "Maricus", se llevará a cabo una comparación entre la versión original y la implementación que hace uso de hilos. Este análisis se basará en métricas específicas de calidad del software, de acuerdo con los criterios establecidos por la norma ISO 25010. Se medirán aspectos clave como el tiempo de ejecución, el consumo de recursos, la estabilidad y la respuesta del sistema.

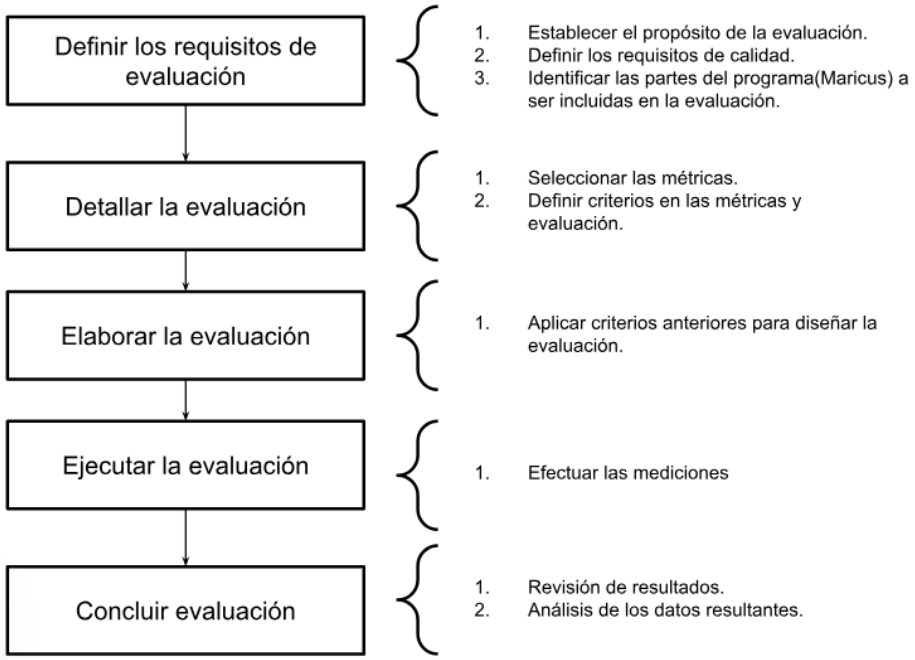
Las pruebas exhaustivas se realizaron en situaciones representativas de carga, que evaluaron el desempeño y la eficiencia de ambas versiones. Se compararán métricas como el tiempo total de ejecución de tareas, la utilización de recursos del sistema y la tolerancia a fallos. Estos datos proporcionarán una perspectiva objetiva sobre la presencia y magnitud de cualquier mejora derivada de la implementación de hilos.

Proceso de Evaluación

Identificar las actividades esenciales y definir los procesos clave para evaluar la calidad, con el empleo de diversas funciones, metas, insumos, resultados, y datos adicionales necesarios en el proceso de evaluación de calidad. En la Figura 8, se evidencia el proceso.

Figura 8

Proceso de Evaluación Aplicado al Maricus



Nota. Expone la secuencia mediante la cual se puede llevar a cabo la evaluación del software, ya sea durante o después del proceso de desarrollo. Adaptado de ISO 25040. (n.d.). https://iso25000.com/index.php/normas-iso-25000/iso-25040

Métricas de Calidad

A través de las siguientes Tablas, se destacan las métricas principales conforme a la norma ISO 25010, enfocadas en evaluar la eficiencia de desempeño y la fiabilidad de Maricus, el programa en cuestión. Estas métricas han sido seleccionadas para proporcionar una evaluación completa de la calidad del software, centrándose específicamente en aspectos como la eficiencia operativa y la confiabilidad del sistema, lo cual se detalla en la Tabla 3 y Tabla 4 para cada parámetro seleccionado de la norma de referencia.

Tabla 3

Subcaracterísticas de rendimiento con sus respectivas métricas

Subcaracterística de rendimiento	Métricas
Comportamiento temporal	Tiempo de respuesta
	Tiempo de espera
Utilización de recursos	Rendimiento
	Utilización de CPU
	Utilización de memoria

Nota. Demuestra las subcaracterísticas par a el rendimiento de acuerdo a la ISO 25010. Recuperado de Orozco Pilco, H. D., & Moreno Tuquinga, S. G. (2021). Desarrollo de una aplicación para dispositivos Android con Geolocalización basado en realidad aumentada para el Sector turístico de la Ciudad de Riobamba (Bachelor's thesis, Riobamba, Universidad Nacional de Chimborazo).

Tabla 4

Subcaracterísticas de fiabilidad con sus respectivas métricas

Subcaracterística de fiabilidad	Métricas
Madurez	Tiempo medio entre fallos
	Eliminación de errores
Disponibilidad	Tiempo de actividad del sistema
Tolerancia a fallos	Anulación de la operación incorrecta

Nota. Demuestra las subcaracterísticas para la fiabilidad de acuerdo a la ISO 25010. Recuperado de Orozco Pilco, H. D., & Moreno Tuquinga, S. G. (2021). Desarrollo de una aplicación para dispositivos Android con Geolocalización basado en realidad aumentada para el

Sector turístico de la Ciudad de Riobamba (Bachelor's thesis, Riobamba, Universidad Nacional de Chimborazo).

Ponderación

La Tabla 5, alineada con la normativa ISO 25010, desempeña un papel esencial en la evaluación de calidad, al asignar pesos y prioridades a los parámetros comparativos. Su diseño sigue los principios de fiabilidad y eficiencia, que proporcionan una estructura organizada para decisiones informadas y una visión detallada de la calidad del software, que respalda la mejora continua.

Tabla 5

Ponderación de Importancia

Nivel	Símbolo	Porcentaje
ALTO	A	80 - 100
MEDIO	B	40 - 79
BAJO	C	1 – 39
No Aplica	NA	0

Nota. Demuestra la ponderación con la que se calificará el desempeño de Maricus en su versión actual como la versión final. Tomado parcialmente de Ortega Moreno, M. (2018). Sistemas de evaluación de la calidad de los componentes web centrado en los usuarios finales (Doctoral dissertation, ETSI_Informatica).

Métricas

Se empieza por establecer las métricas para las subcaracterísticas que comprende la evaluación de la eficiencia del desempeño. A continuación, se analizó detalladamente cada métrica para obtener una comprensión más precisa de su impacto en el rendimiento general del sistema.

Eficiencia de desempeño. La evaluación del rendimiento se lleva a cabo comúnmente durante la fase de pruebas en el desarrollo de software. Sin embargo, para asegurar un entorno fidedigno y resultados sólidos, se ha decidido implementar esta evaluación en la versión en producción de Maricus, un programa preexistente en la empresa MYM. A continuación, se presentan las métricas de rendimiento y las fórmulas que se utilizarán en este proceso de evaluación, se destaca la importancia de extender el análisis más allá de la fase de desarrollo para garantizar la confiabilidad y validez de los resultados obtenidos en un entorno operativo real.

Desempeño temporal. Esta faceta incluye los lapsos de respuesta y procesamiento de una aplicación, Maricus, durante su ejecución en circunstancias específicas. La evaluación de esta característica se realiza mediante tres medidas: i) intervalo de respuesta, ii) periodo de espera, iii) eficiencia en un periodo determinado. Para recopilar datos de todas estas mediciones, se hizo uso de los logs existentes que detallan la hora de inicio y fin del programa que facilita la captura de diversos lapsos parciales de manera más exacta. Las métricas con sus fórmulas correspondientes se pueden ver en la Tabla 6.

Tabla 6

Métricas del desempeño temporal

Métrica	Fórmula	Descripción
Tiempo de respuesta	X = Z-N	N=Tiempo de envío de la petición Z=Tiempo de espera para recibir la primera respuesta
Tiempo de espera	X=Z-N	N=Tiempo inicial de una tarea Z=Tiempo final de la tarea
Rendimiento	X=N/W	N=Tareas completadas W=Tiempo

Nota. Muestra las fórmulas correspondientes a cada métrica del desempeño temporal. Tomado parcialmente de Carrión Vaca, G. E. (2018). Comparativa de tres herramientas de realidad aumentada utilizando una metodología de medición de software ISO 25010 (Bachelor's thesis).

Aprovechamiento de recursos. Las métricas relacionadas con el aprovechamiento de recursos proporcionan una valiosa herramienta para evaluar la eficiencia en el uso de los recursos por parte del software. En el marco de la normativa ISO/IEC 25010, se destaca la importancia de medir y analizar cómo el programa gestiona y utiliza sus recursos. En la Tabla 7, se presentan detalladamente estas métricas, acompañadas de las fórmulas correspondientes, lo que ofrece así una visión integral de la optimización de recursos en el contexto de la evaluación de la calidad del software.

Tabla 7

Métricas del aprovechamiento de recursos

Métrica	Fórmula	Descripción
Utilización de CPU	$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$	Ai= Porcentaje de procesador realmente utilizado. Bi= Porcentaje de procesador disponible n = Número de observaciones
Utilización de memoria	$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$	Ai= Tamaño de la memoria realmente utilizada para realizar un determinado conjunto de tareas. Bi= Tamaño de la memoria disponible para realizar las tareas durante el

Métrica	Fórmula	Descripción
		procesamiento n = Número de observaciones

Nota. Se muestra las métricas del aprovechamiento de recursos con sus correspondientes fórmulas. Tomado parcialmente de Carrión Vaca, G. E. (2018). Comparativa de tres herramientas de realidad aumentada utilizando una metodología de medición de software ISO 25010 (Bachelor's thesis).

Fiabilidad. La fiabilidad en la evaluación del software, conforme a los estándares de la norma ISO 25010, se torna fundamental para asegurar un desempeño consistente y confiable en un entorno operativo real. En este contexto, se implementa la evaluación de fiabilidad en la versión en producción del software, lo que garantiza así resultados sólidos y válidos en condiciones reales. Se destacan métricas clave, como el tiempo promedio entre fallos (MTBF) y el tiempo promedio de reparación (MTTR), subraya la importancia de prevenir fallos mediante estrategias proactivas y la aplicación de monitoreo continuo para identificar y abordar posibles problemas antes de que impacten significativamente la confiabilidad del software.

Madurez. Dentro del marco de evaluación de la normativa ISO/IEC 25010, la métrica de madurez se centra en medir el grado de desarrollo y estabilidad alcanzado por el software. Esta métrica es esencial para comprender la evolución y la consistencia del programa a lo largo del tiempo. En la tabla 8, se presentan detalladamente las métricas asociadas a la madurez, junto con las fórmulas pertinentes, lo que ofrece así una perspectiva completa de cómo se evalúa y cuantifica este aspecto fundamental en el contexto de la calidad del software.

Tabla 8

Métricas de la madurez

Métrica	Fórmula	Descripción
Tiempo medio entre fallos	X = T/Z	Z= Número total de fallas detectadas actualmente T= Tiempo de operación N>0
Eliminación de errores	X=Z/N	Z= Número de fallas corregidas en la fase de diseño/codificación /pruebas N = Número de fallas detectadas en las pruebas N>0

Nota. Se muestra las métricas de madurez con sus correspondientes fórmulas. Tomado parcialmente de Carrión Vaca, G. E. (2018). Comparativa de tres herramientas de realidad aumentada utilizando una metodología de medición de software ISO 25010 (Bachelor's thesis).

Disponibilidad. La métrica de disponibilidad se focaliza en evaluar la capacidad del software para estar accesible y operativo cuando se requiere. Esta métrica es crucial para medir la fiabilidad y accesibilidad del programa en diferentes situaciones. A continuación, se detallan las métricas asociadas a la disponibilidad, acompañadas de sus respectivas fórmulas, proporciona así una visión integral de cómo se evalúa y cuantifica este aspecto esencial en el ámbito de la calidad del software.

Tabla 9

Métricas de la disponibilidad

Métrica	Fórmula	Descripción
Tiempo de actividad del sistema	X = Z/N	Z=Tiempo del servicio proporcionado actualmente N=Tiempo de servicio del sistema regulado en el cronograma operacional N>0
Tiempo promedio de inactividad del sistema	X=Z/T	Z=Número total de fallas detectadas actualmente T=Tiempo de la inactividad T>0

Nota. Se muestra las métricas de disponibilidad con sus correspondientes fórmulas. Tomado parcialmente de Carrión Vaca, G. E. (2018). Comparativa de tres herramientas de realidad aumentada utilizando una metodología de medición de software ISO 25010 (Bachelor's thesis).

Tolerancia a fallos. La métrica de tolerancia a fallos se centra en evaluar la capacidad del software para mantener un desempeño aceptable incluso en presencia de fallos o situaciones inesperadas. Esta métrica es esencial para medir la robustez y la capacidad de recuperación del programa frente a eventos imprevistos. A continuación, se presentan detalladamente las métricas asociadas a la tolerancia a fallos, acompañadas de sus respectivas fórmulas, lo que ofrece así una visión completa de cómo se evalúa y cuantifica este aspecto crítico en el ámbito de la calidad del software.

Tabla 10

Métrica de la Tolerancia a Fallos

Métrica	Fórmula	Descripción
Anulación de la operación incorrecta	X = Z/N	Z=Número de operaciones incorrectas encontradas N=Número de funciones implementadas para anular las operaciones incorrectas N>0

Nota. Se muestra las métricas de tolerancia a fallas con sus correspondientes fórmulas. Tomado parcialmente de Rivera Guaraca, M. I. (2021). Evaluación de la fiabilidad en el sistema web de agendamiento de citas médicas del Hospital General Universitario Andino de la Provincia de Chimborazo (Bachelor's thesis, Riobamba, Universidad Nacional de Chimborazo).

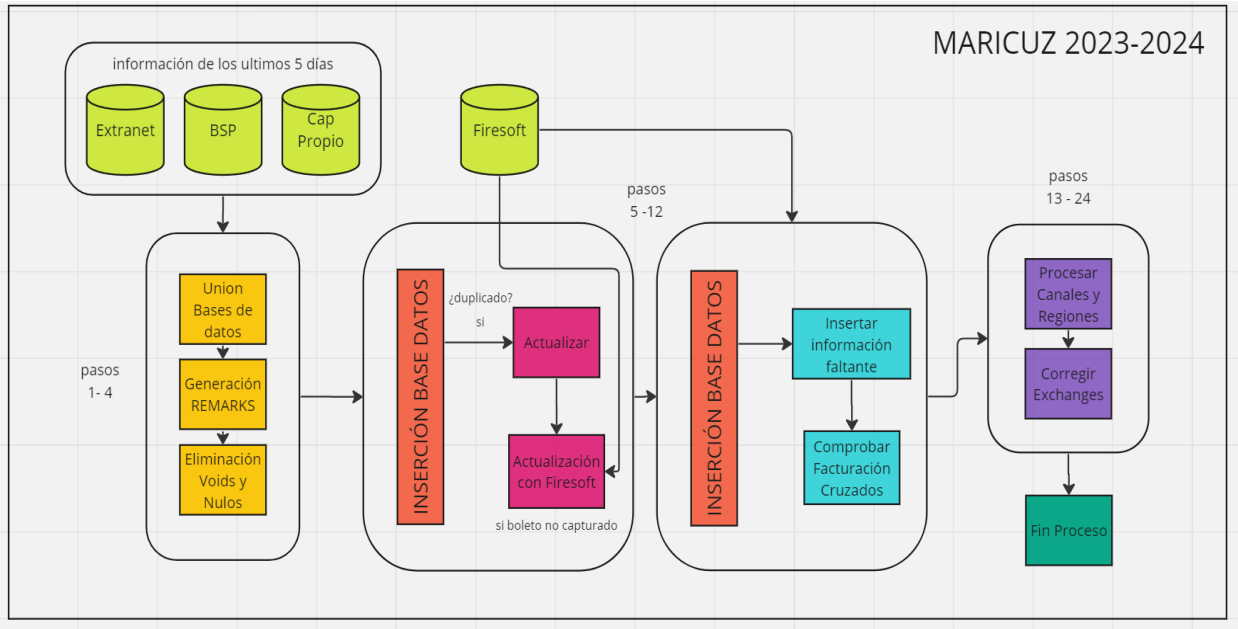
Condiciones iniciales para la evaluación

Para una comprensión más profunda de la situación actual y el proceso ETL en Maricus, se utiliza la Figura 9, que ilustra las bases de recopilación de información y su procesamiento a través de los pasos definidos en el sistema. En esta representación visual, se evidencia de manera clara el flujo de datos y las distintas etapas del proceso ETL, lo cual

brinda una perspectiva que facilita la comprensión de las condiciones iniciales para la evaluación de Maricus y su sistema de gestión de datos.

Figura 9

Proceso ETL de Maricus



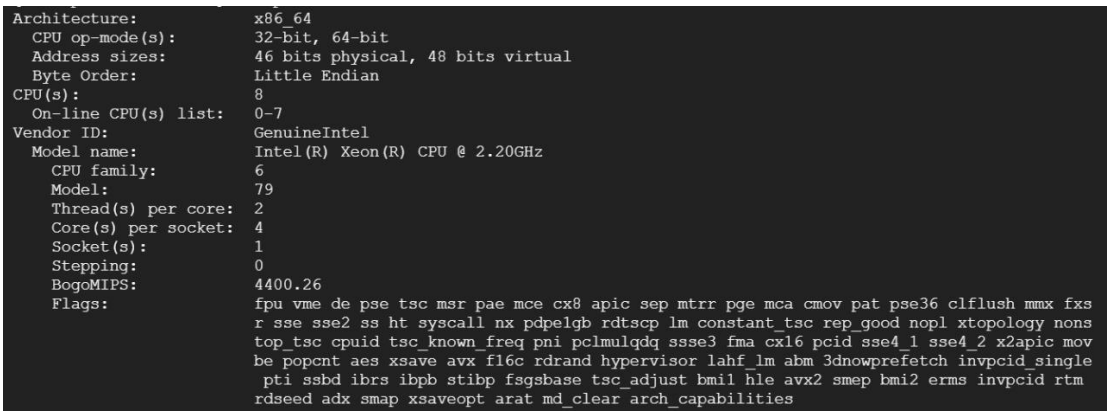
Nota. Demuestra los pasos del proceso ETL presente en el programa Maricus. Autoría Propia

Maricus se encuentra actualmente en la computadora detallada en la tabla 13, se somete a un análisis según la norma ISO 25010. En este contexto, se evalúan las métricas de desempeño temporal y fiabilidad, para lo cual se considera las características específicas del ordenador que ejecuta las aplicaciones. Los detalles iniciales de este sistema informático se encuentran en la tabla adjunta, que proporcionan una base para la evaluación. Este enfoque permite comprender el rendimiento de Maricus y su adaptación al entorno computacional. La tabla actúa como referencia para la evaluación conforme a los estándares de la ISO 25010, que presenta las condiciones originales. Este análisis integral ofrece una comprensión detallada de cómo Maricus utiliza los recursos y asegurar un estudio exhaustivo de su rendimiento. Por ello

en la Tabla 11 se da a conocer las características del procesador en el que corre Maricus y en la Figura 10 se muestra la información adicional del servidor.

Figura 10

Características del servidor



Nota. La figura muestra información adicional de relevancia del servidor, de autoría propia

Tabla 11

Condiciones Iniciales del computador

Condiciones iniciales del computador	
Sistema operativo	Centos 07
CPU	8 vCPU, 4 core
RAM	64 GB
PROCESADOR	INTEL(R) Xeon(R)

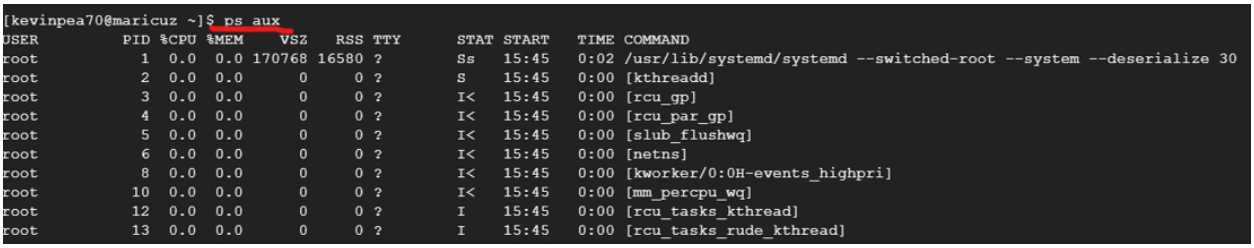
Nota. La tabla muestra las características del *servidor* que aloja actualmente al programa Maricus, de autoría propia

Como se menciona en el Apéndice 4. Para observar las características del computador, se ejecuta el comando ps aux, que se observa en la Figura 11, que permite verificar de manera

general la arquitectura, CPU, modelo, entre otros detalles propios de la máquina, así como se muestra en la siguiente imagen.

Figura 11

Uso de comandos para acceder a las características del computador

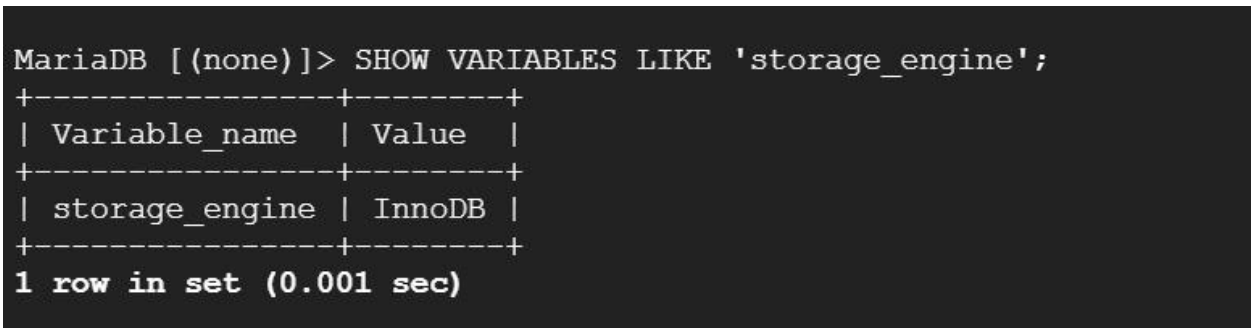


Nota. La imagen muestra el comando usado para visualizar las características del computador que aloja Maricus, la imagen es una captura de autoría propia.

En cuanto a las condiciones iniciales de la base de datos denominada “Repoboleto”, la cual recopila los registros provenientes de la ejecución de Maricus, se puede visualizar el tipo de motor de almacenamiento con ayuda del comando que se visualiza en la Figura 12, que maneja actualmente la base de datos, para este caso es InnoDB.

Figura 12

Motor de almacenamiento de la Base de Datos



Nota. Demuestra el tipo de motor de almacenamiento de la base de datos. Autoría Propia

Al hablar de la base de datos es importante introducir el concepto de Big Data, al referirnos a esta en la empresa, se debe hacer referencia a las 5V definidas con anterioridad en el marco teórico, las cuales de acuerdo con la data de la empresa son:

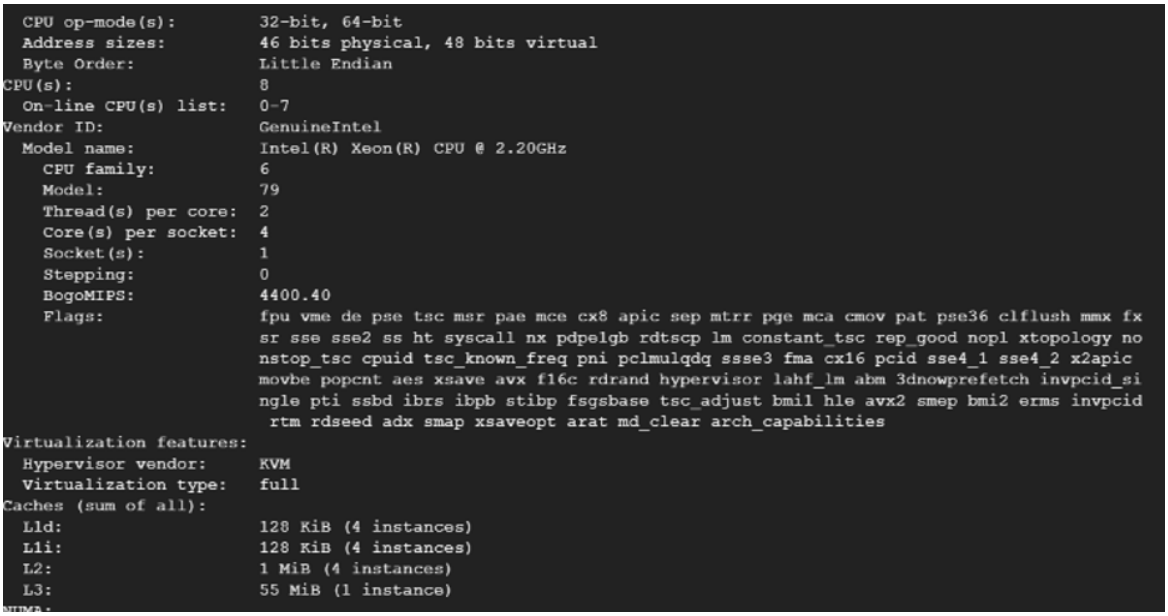
- Volumen: en este caso se manejan en promedio 46000 datos mensuales.
- Velocidad: tarda alrededor de dos horas para procesar 15000 registros.
- Variedad: la data de la empresa representa una gran variedad de datos, ya que la misma es recolectada de tres distintas bases. Adicional a esto la tabla dónde se junta la Big Data cuenta con 94 campos.
- Valor: gracias a la Big Data, la empresa puede tomar decisiones acertadas que agilitan los procesos en la empresa y generan capacidad competitiva.
- Veracidad: la información de la Big Data se carga a diario.

Unidad de análisis

Para la unidad de análisis, se optó por un muestreo selectivo, se usan los datos de boletos registrados en los últimos seis meses a través de los registros del programa MARICUS y el monitoreo del sistema. Los resultados obtenidos serán utilizados para evaluar el grado de confiabilidad del sistema web, conforme a los criterios establecidos en la norma de calidad ISO/IEC 25010.

Figura 13

Características del Servidor



Nota. Demuestra las características e información propia del servidor en el que se ejecuta Maricus. Autoría Propia

Población de estudio. Al ejecutar el programa Maricus, se establece en la fecha de entrada el 01/01/2023 y como fecha de salida 31/12/2023; es así que se obtiene un total de 158155 registros de boletos como se visualiza en la Figura 14, cabe recalcar que la empresa se recupera de una temporada post pandemia. Ese es el total de registros que van a ser insertados o actualizados que dependen de las condiciones establecidas en el resto de pasos del programa; ya que como se ha mencionado Maricus en una Big Data, producto de varias bases de datos.

Figura 14

Total, de boletos recogidos en la Base de Datos BSP

Nota. La imagen muestra el número de registros existentes en el año dentro de la BBDD denominada BSP, la imagen es una captura de autoría propia

Tamaño de la muestra. La muestra utilizada para el análisis de los parámetros de la normativa ISPO 25010 se extrajo de forma representativa de la población total. Se aplicó la fórmula correspondiente, se seleccionó un conjunto significativo de datos que refleja con precisión la diversidad y características de la población. En este caso, la muestra comprende los datos de los boletos emitidos durante un mes por la empresa Casa de Incentivos Casintour. Estos datos servirán como base para realizar un análisis detallado, lo que permite evaluar y mejorar la calidad de los servicios ofrecidos por la empresa de acuerdo con los estándares establecidos por la normativa ISPO 25010.

En el cálculo de la muestra, se ha considerado el nivel de confianza deseado y el margen de error aceptable, elementos esenciales para garantizar la representatividad y validez

de los resultados. La fórmula empleada para determinar el tamaño de la muestra ha tenido en cuenta estos parámetros, los cuales se encuentran detallados en las Tablas 12 y 14.

Tabla 12

Margen de error para fórmula de la muestra

Margen de error	**e**
1%	0,01
2%	0,02
3%	0,03
4%	0,04
5%	0,05
9%	0,09

Nota. La tabla muestra los valores para el margen de error. Recuperada de Aguilar-Barojas, S. (2005). Fórmulas para el cálculo de la muestra en investigaciones de salud. Salud en tabasco, 11(1-2), 333-338.

El nivel de confianza, esencial en la obtención de muestras, representa la seguridad en que un intervalo de confianza englobe el parámetro de interés de la población. La importancia de esta medida se destaca con claridad en la Tabla 13, donde se proporciona un análisis detallado al respecto. Además, este aspecto es fundamental para la validez y precisión de los resultados obtenidos.

Tabla 13

Nivel de confianza para aplicar a la fórmula de la muestra

Nivel de Confianza	z
99%	2,58
98%	2,53
97%	2,17
96%	2,05
95%	1,96
90%	1,65

Nota. La tabla muestra los valores para el nivel de confianza. Recuperada de Aguilar-Barojas, S. (2005). Fórmulas para el cálculo de la muestra en investigaciones de salud. Salud en tabasco, 11(1-2), 333-338.

Tabla 14

Probabilidad de éxito o fracaso

Valor de “p” y “q”		
Probabilidad de éxito = p	50%	0,5
Probabilidad de fracaso = q	50%	0,5

Nota. La tabla muestra los valores para el Probabilidad de éxito o fracaso. Recuperada de Aguilar-Barojas, S. (2005). Fórmulas para el cálculo de la muestra en investigaciones de salud. Salud en tabasco, 11(1-2), 333-338.

Una vez con los datos se aplica la formula y si se trabaja con una población finita y cse conoce el tamaño total de la población (N), la fórmula ajustada para el tamaño de muestra es la siguiente:

$$n = \frac{N.Z^2.p.(1-p)}{N.E^2 + Z^2.p(1-p)}$$

Dado que se conoce $\mathrm{N} = 158155$, $Z = 1.96$, $p = 0.5$, y se asume un margen de error $E = 0.01$, se puede sustituir estos valores en la fórmula:

$$n = \frac{158155.(1.96)^2.0.5.(1-0.25)}{158155.(0.01)^2 + (1.96)^2.0.5(1-0.5)}$$

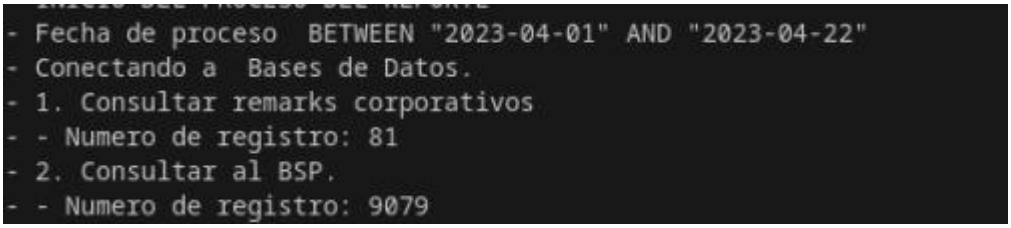
$$n = 9054.182$$

Después de aplicar la fórmula para determinar el tamaño de la muestra, se obtiene un resultado cercano a 9054.182 registros. Con este número en mente, se procede a buscar en Maricus un mes que cumpla con las condiciones requeridas por la muestra, lo que asegura así una representación adecuada para el análisis que se llevará a cabo.

Tras evaluar los registros disponibles en Maricus, se determinó que el mes que se aproximó de manera más significativa al tamaño de la muestra calculado es el mes 4, es decir, abril del 2023, como se logra ver en la Figura 15. En este periodo, se registraron un total de 9079 boletos, lo que lo convierte en la elección más adecuada para llevar a cabo el análisis correspondiente, ya que se ajusta de manera cercana a la muestra requerida.

Figura 15

Muestra obtenida de la ejecución de Maricus



Nota. La imagen comprende la muestra obtenida de la ejecución de Maricus en el cuarto mes del año 2023. La imagen es una captura de un entorno de ejecución, autoría propia.

Técnicas de Recolección de Datos. Método de observación: Se empleó la información recopilada a partir de los registros generados por el programa Maricus, en combinación con los comandos que proporciona Linux como vmstat y mpstat. Estos comandos posibilitaron la evaluación del desempeño de componentes clave, tales como la CPU y la RAM, junto con otras características relevantes. Este enfoque de observación detallada contribuyó de manera integral a la valoración exhaustiva de los aspectos fundamentales del sistema.

Análisis e Interpretación de la Información. La evaluación de la investigación se llevó a cabo mediante el análisis de métricas relacionadas con diversos subcriterios, como madurez, disponibilidad y tolerancia a fallos, pertenecientes al criterio de fiabilidad. Asimismo, se consideraron aspectos como el comportamiento temporal y la utilización de recursos en el ámbito del criterio de eficiencia de desempeño.

Estos parámetros están detallados en el modelo de calidad ISO/IEC 25010. Para llevar a cabo la evaluación, se utilizó la información de los registros del programa Maricus, junto con la herramienta de monitoreo integrada en Linux, que permite verificar el rendimiento de elementos como la CPU y la RAM, entre otras características. Se aplicó un modelo con una

escala de valoración para evaluar la calidad del producto de software, que representa así el nivel global de satisfacción con respecto al cumplimiento de los requisitos de calidad establecidos por la norma ISO/IEC.

Aplicación de las Pruebas

En la siguiente imagen se exhiben los procesos actualmente en ejecución en el servidor que alberga el programa "Maricus". Cabe destacar que estos procesos son exclusivamente inherentes al sistema operativo, sin la presencia de tareas adicionales o programas de usuario.

La visualización detallada de estos procesos proporciona una instantánea precisa de las operaciones internas esenciales para el funcionamiento básico del sistema, destaca aspectos como la gestión de memoria, el sistema de archivos y otras funciones fundamentales del sistema operativo. Este enfoque en los procesos propios del sistema permite una evaluación directa del rendimiento del sistema en su estado fundamental, procesos que pueden ser vistos en la Figura 16.

Figura 16

Procesos propios del computador

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 170812 16416 ?        Ss   Feb19   0:08 /usr/lib/systemd/systemd --switched-root --sy
root           2  0.0  0.0      0     0 ?        S    Feb19   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Feb19   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Feb19   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   Feb19   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   Feb19   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   Feb19   0:00 [kworker/0:0H-events_highpri]
root          10  0.0  0.0      0     0 ?        I<   Feb19   0:00 [mm_percpu_wq]
root          12  0.0  0.0      0     0 ?        I    Feb19   0:00 [rcu_tasks_kthread]
root          13  0.0  0.0      0     0 ?        I    Feb19   0:00 [rcu_tasks_rude_kthread]
root          14  0.0  0.0      0     0 ?        I    Feb19   0:00 [rcu_tasks_trace_kthread]
root          15  0.0  0.0      0     0 ?        S    Feb19   0:02 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        I    Feb19   0:25 [rcu_preempt]
root          17  0.0  0.0      0     0 ?        S    Feb19   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    Feb19   0:00 [cpuhp/0]
root          20  0.0  0.0      0     0 ?        S    Feb19   0:00 [cpuhp/1]
root          21  0.0  0.0      0     0 ?        S    Feb19   0:00 [migration/1]
root          22  0.0  0.0      0     0 ?        S    Feb19   0:02 [ksoftirqd/1]
root          24  0.0  0.0      0     0 ?        I<   Feb19   0:00 [kworker/1:0H-events_highpri]
root          25  0.0  0.0      0     0 ?        S    Feb19   0:00 [cpuhp/2]
root          26  0.0  0.0      0     0 ?        S    Feb19   0:00 [migration/2]
root          27  0.0  0.0      0     0 ?        S    Feb19   0:01 [ksoftirqd/2]
root          29  0.0  0.0      0     0 ?        I<   Feb19   0:00 [kworker/2:0H-events_highpri]
```

Nota. La imagen muestra los procesos propios del servidor en el que corre el programa Maricus. La imagen es una captura de un entorno de ejecución, autoría propia.

Tras examinar los procesos a través de la terminal de Linux en el sistema que aloja el programa "Maricus", se revela que los procesadores operan a niveles mínimos de carga cuando se ejecuta el programa. Asimismo, se observa un consumo de memoria significativamente bajo. Estos indicadores sugieren que, en el momento de la revisión, el sistema se encuentra en un estado de eficiencia y gestión óptima de recursos.

Demostración del Proceso de Selección de la Población

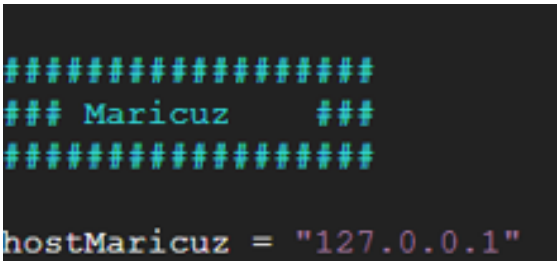
Dentro del proceso de demostración de la población para el análisis de Maricus, se llevará a cabo una revisión del número total de registros recopilados durante el extenso periodo del 01/01/2023 al 31/12/2023. Esta verificación es esencial para garantizar la amplitud y representatividad de la muestra. Posteriormente, se hace uso de esta población completa, se seleccionó estratégicamente una muestra que aborde distintos aspectos del software, desde su funcionalidad hasta su eficiencia, confiabilidad, usabilidad, y demás criterios definidos por la norma ISO 25010.

Esta muestra se convertirá en el foco de análisis exhaustivos, lo que permite una evaluación detallada del desempeño del programa. Este enfoque sistemático y normativo asegura que la evaluación cumpla con estándares de calidad reconocidos, contribuye a la identificación de áreas de mejora y optimización en el software Maricus.

Dentro del proceso para determinar el número total de registros en la población, se accederá a una base de pruebas específica con el uso de las credenciales correspondientes, como en la Figura 17, para garantizar un acceso autorizado y seguro a la información. Se deben ingresar las fechas de entrada y salida, para que se realice el proceso ETL de Maricus.

Figura 17

Base de datos a la que apunta Maricus

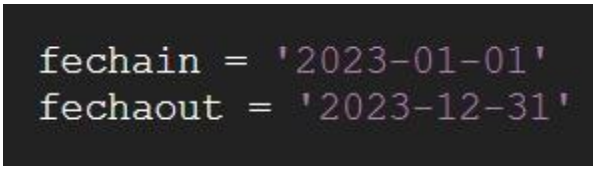


Nota. La imagen comprende la base de datos Maricus. La imagen es una captura de un entorno de ejecución, autoría propia.

Al proporcionar estas fechas, se pretende delimitar el rango temporal y obtener así una visión precisa de los registros generados durante el periodo seleccionado, que abarca desde el 01/01/2023 hasta el 28/12/2023, como se puede ver en la Figura 18. Este enfoque meticuloso asegura que la información recopilada sea coherente con los parámetros establecidos para la evaluación.

Figura 18

Fecha de entrada y salida para la ejecución de Maricus

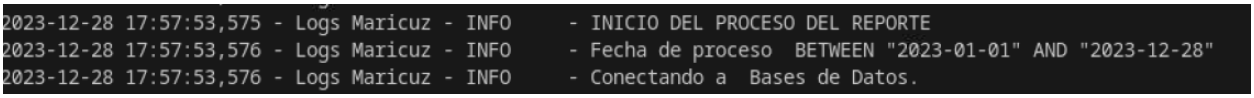


Nota. Se pueden observar las fechas usadas para la ejecución de Maricus. La imagen es una captura de un entorno de ejecución, autoría propia.

El inicio de la fase de procesamiento implica la extracción de un año completo de datos, y para llevar a cabo este proceso, se inicia con la consulta a las bases de datos BSP y CAPTURADOR, se puede observar la ejecución de ese paso en la Figura 19.

Figura 19

Programa Maricus en ejecución



Nota. La imagen muestra Maricus en ejecución con el intervalo de fechas ingresadas. La imagen es una captura de un entorno de ejecución, autoría propia.

Tras finalizar el proceso de verificación de la población y la subsiguiente selección de la muestra, se procede a realizar las pruebas necesarias sobre este conjunto específico de datos. Estas pruebas tienen como objetivo determinar el estado actual del programa Maricus, así también se consideran aspectos clave de su desempeño y funcionalidad. Este análisis minucioso permitió comparar el rendimiento de la versión actual de Maricus con la versión mejorada, que incorpora técnicas avanzadas de paralelismo basado en procesos. Para el registro de dichos datos, el programa corrió durante un tiempo aproximado de 3 horas.

Inicio de Análisis

Para dar inicio al análisis, se han desarrollado scripts específicos que, en colaboración con los registros proporcionados por el programa Maricus, desempeñarán un papel fundamental en la recopilación de datos relacionados con el tiempo y el uso del CPU. Estos scripts están diseñados con el propósito de brindar una comprensión más profunda y detallada de los patrones de rendimiento, alineándose con los parámetros previamente establecidos en

las tablas que detallan las subcaracterísticas de cada componente según la norma ISO 25010, centrada particularmente en la eficiencia de desempeño y la fiabilidad.

La utilización de estos scripts y registros permitirá una evaluación precisa y detallada, que proporcionan así una base sólida para el análisis subsecuente y la identificación de posibles áreas de mejora dentro del sistema.

En la situación inicial de Maricus, se observa que la demanda principal consiste en procesar un año completo de registros para la generación de informes internos dentro de la empresa. Este escenario impone una carga significativa en términos de tiempo y recursos del CPU. Además, es relevante destacar que el programa Maricus se ejecuta específicamente en horarios de baja actividad, 5 AM, la hora predeterminada, ya que se aprovecha la disponibilidad del servidor en ese periodo.

Esta elección de horario se realiza con la intención de minimizar el impacto en las operaciones diurnas, lo que garantiza que la información esté lista y disponible para su uso durante las horas de trabajo más críticas. Este contexto temporal añade una capa adicional de complejidad al análisis, ya que es crucial evaluar el rendimiento del sistema bajo condiciones específicas y considerar la eficiencia de procesamiento durante las horas pico y fuera de ellas.

A continuación, se procederá con el análisis individual de las subcaracterísticas de eficiencia de desempeño y fiabilidad, conforme a los estándares establecidos por la norma ISO 25010. Para evaluar la eficiencia de desempeño, se enfocará en aspectos como la velocidad de procesamiento de los datos y la capacidad de respuesta del sistema, además, se consideran los patrones de uso del CPU y la optimización de los scripts desarrollados.

Paralelamente, se abordará la fiabilidad, que examina la consistencia en la generación de informes y la capacidad del programa para mantener un rendimiento consistente a lo largo del tiempo. Este análisis detallado permitirá identificar áreas de mejora específicas y

proporcionará una visión detallada de la eficacia del sistema Maricus en relación con las subcaracterísticas mencionadas.

Comportamiento Temporal.

Tiempo de Respuesta. En cuanto al tiempo de respuesta, al ejecutar Maricus, se ha registrado un tiempo promedio de 18 segundos para el envío de peticiones a la base de datos (BSP y Capturador). Cabe destacar que, en caso de no recibir respuesta de la base de datos, Maricus se detiene automáticamente. Esta interrupción se produce debido a que las peticiones a las bases de datos están configuradas para cerrar todas las conexiones en ausencia de una respuesta.

$$X = Z - N$$

$$X = \ 19{:}21{:}09 - \ 19{:}20{:}51$$

$$X = 18\ seg$$

En la versión optimizada de Maricus, se ha priorizado la minimización del tiempo de respuesta, reduciéndolo al máximo. En este contexto, si el programa no recibe una respuesta de la base de datos BSP y/o Capturador, sigue con la medida de detenerse automáticamente. Esta estrategia se mantiene para evitar posibles demoras innecesarias en la ejecución del sistema, y para asegurar que las operaciones se realicen de manera eficiente y dentro de límites temporales aceptables.

$$X = Z - N$$

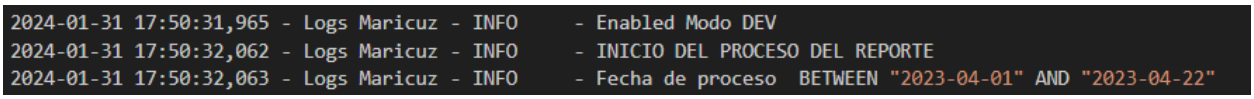
$$X = \ 18{:}45{:}07 - \ 18{:}44{:}54$$

$$X = \quad 13\ seg$$

Tiempo de Espera. En lo que respecta al tiempo de espera, se ha considerado cada paso como una tarea dentro del flujo de Maricus. No obstante, para evaluar de manera más integral el tiempo de espera, se enfocó en la ejecución completa del programa. A través de los registros internos del programa, se ha determinado la hora de inicio y la hora de finalización al procesar, como se muestra en las Figuras 20 y 21, respectivamente. En este análisis, para la prueba se va a hacer uso de la muestra que corresponde a un mes aproximadamente.

Figura 20

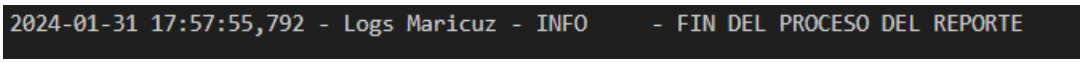
Log de Información del proceso de inicio de Maricus Original



Nota. La imagen comprende la hora de inicio de la ejecución de Maricus. La imagen es una captura de un entorno de ejecución, autoría propia.

Figura 21

Log de información del fin de proceso de Maricus Original



Nota. La imagen comprende la hora de fin de la ejecución de Maricus. La imagen es una captura de un entorno de ejecución, autoría propia.

Dada la prueba que se efectuó sobre el mes de abril con 9074 registros, por insertar o actualizar de acuerdo a las normas del programa se obtuvo que:

Hora de fin: 17:57:55

Hora de inicio: 17:50:31

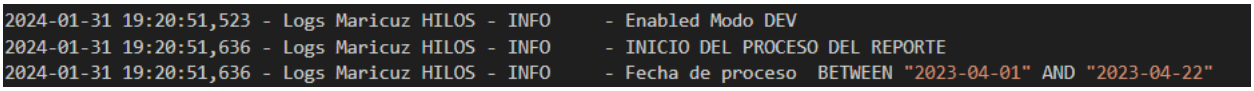
Por lo cual, se aplica la fórmula correspondiente a esta submétrica X = Z-N, donde N es

el tiempo inicial de la tarea y Z el tiempo final de la tarea, $X = 7{:}24$

Al someter el programa optimizado de Maricus, en el cual se han implementado hilos para mejorar la eficiencia del procesamiento, se ha evidenciado una notable reducción en el tiempo de espera. De igual manera, se aplica la misma medición sobre la muestra y se obtienen los logs que evidencian la hora de inicio y la hora final, como se muestra en las Figura 22 y Figura 23, respectivamente.

Figura 22

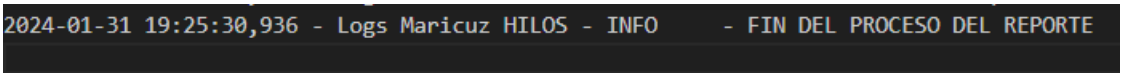
Log de información del proceso de Inicio de Maricus con Hilos



Nota. La imagen comprende la hora de inicio de la ejecución de Maricus con Hilos. La imagen es una captura de un entorno de ejecución, autoría propia.

Figura 23

Log de información del proceso final de Maricus con Hilos



Nota. La imagen comprende la hora final de la ejecución de Maricus con Hilos. La imagen es una captura de un entorno de ejecución, autoría propia.

Hora de fin: 19:25:30

Hora de inicio: 19:20:51

Por lo cual se aplica la fórmula correspondiente a esta submétrica X = Z-N, donde N es el tiempo inicial de la tarea y Z el tiempo final de la tarea, X= 4:39

Para calcular el porcentaje de reducción del tiempo, se usa la siguiente fórmula:

$$Porcentaje\ de\ reducción = \left(\frac{Tiempo\ anterior - Tiempo\ actual}{Tiempo\ anterior}\right) * 100$$

Dado que el tiempo anterior era 7 minutos y 24 segundos, y el tiempo actual es 4 minutos y 39 segundos, puedes sustituir estos valores en la fórmula:

$$Porcentaje\ de\ reducción = \left(\frac{7\ min + 24\ seg - (4\ \text{min} + 39\ seg)}{7\ \text{min} + 24\ seg}\right) * 100$$

Se calcula:

$$Porcentaje\ de\ reducción = \left(\frac{444\ seg - 279\ seg}{444\ seg}\right) * 100 = \left(\frac{165\ seg}{444\ seg}\right) * 100$$

Por lo tanto, el porcentaje de reducción del tiempo es aproximadamente 37.16% Esto indica que el nuevo tiempo es aproximadamente el 37.16% menor que el tiempo anterior.

Rendimiento. Para evaluar el rendimiento de Maricus, se ha adoptado como parámetro la cantidad de tareas completadas, se pone a consideración que cada tarea representa la realización de los 24 pasos que actualmente constituyen el flujo operativo del programa. Con el objetivo de determinar el rendimiento específico de cada paso, se ha medido el tiempo de ejecución de manera individual, de tal forma que se mantiene un enfoque similar al utilizado para evaluar el tiempo total de procesamiento del programa. Este análisis detallado por tarea proporciona información valiosa sobre la eficiencia de cada componente de Maricus, lo que permite identificar posibles cuellos de botella y áreas específicas que podrían beneficiarse de optimizaciones adicionales para mejorar el rendimiento general del sistema.

$$X = \frac{N}{W}$$

$$X = \frac{24}{07{:}\,24min}$$

De manera análoga, el análisis detallado del rendimiento se aplicó al programa optimizado de Maricus, donde se implementaron mejoras significativas, esto incluye el uso de hilos para una ejecución más eficiente. Al evaluar la cantidad de tareas completadas, se observó una reducción significativa en el tiempo de ejecución individual de cada paso. Este enfoque detallado permitió identificar áreas específicas dentro del flujo de Maricus que se beneficiaron especialmente de las optimizaciones, lo que contribuye así a la notable disminución en el tiempo total de procesamiento.

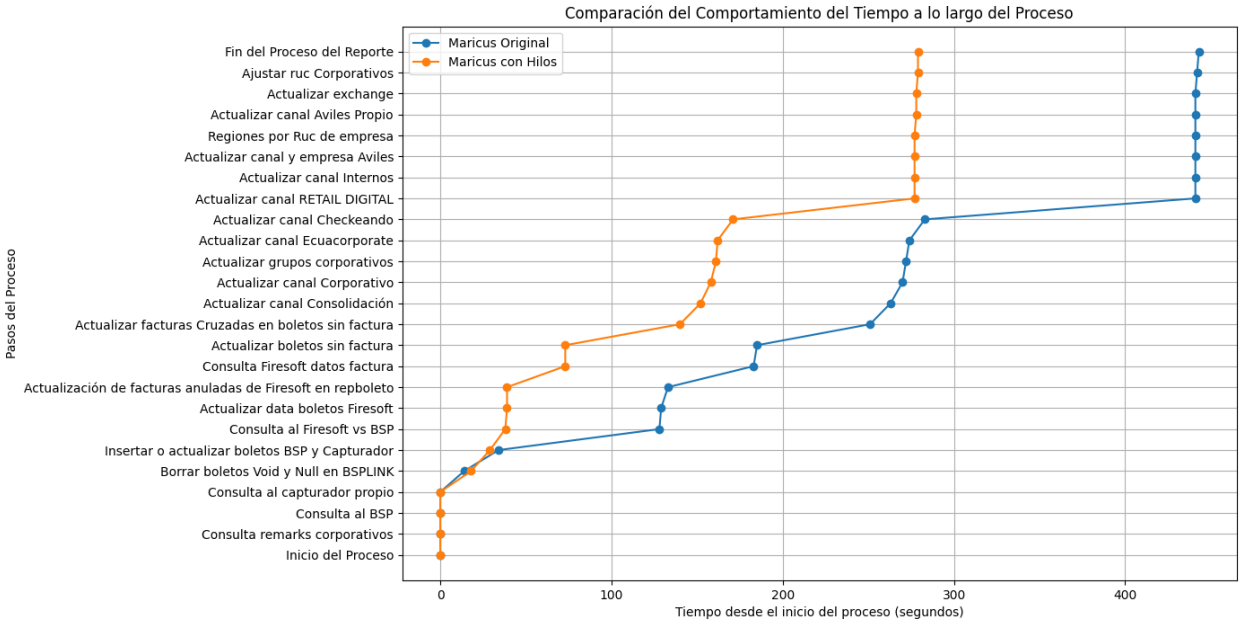
$$X = \frac{N}{W}$$

$$X = \frac{24}{4{:}\,39\ min}$$

En la Figura 24 se puede observar a detalle la comparación del comportamiento temporal a lo largo del proceso, es decir, el paso a paso tanto del programa original como el programa con hilos, en lo que se puede visualizar cuanto tarda cada paso en ir al siguiente.

Figura 24

Comportamiento temporal de Maricus Original y Maricus con Hilos



Nota. La imagen muestra el comportamiento temporal por pasos tanto de Maricus Original como con Hilos. La imagen se realizó en Colab, autoría propia.

Utilización de recursos.

Utilización de CPU. En cuanto al aprovechamiento de recursos, específicamente la medición de la utilización de la CPU, se ha realizado una evaluación que considera la ejecución total del programa Maricus. Para recopilar información detallada sobre el comportamiento del CPU, se implementó un script dedicado que registró datos relevantes durante la ejecución del programa. Los resultados obtenidos indican que, en promedio, el uso del CPU oscila entre el 6% y 7%, como se muestra en la siguiente fórmula:

$$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$$

$$X = 0.073$$

Promedio de CPU Usado para Maricus Original: 6.41%

Con la implementación del programa optimizado de Maricus, se llevó a cabo una nueva medición de la utilización de la CPU, se usó el mismo script para la recolección de datos. Los resultados revelaron un incremento esperado en la utilización del CPU. Este aumento en la carga del CPU indica un mejor aprovechamiento de los recursos disponibles, por lo cual se evidencia que las optimizaciones implementadas impulsan una mayor eficiencia en términos de procesamiento y rendimiento del sistema. Este ajuste en la utilización de la CPU sugiere una optimización más efectiva de los procesos internos de Maricus, se respaldó la idea de que las modificaciones realizadas han conducido a una gestión más eficiente de los recursos del sistema.

$$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$$

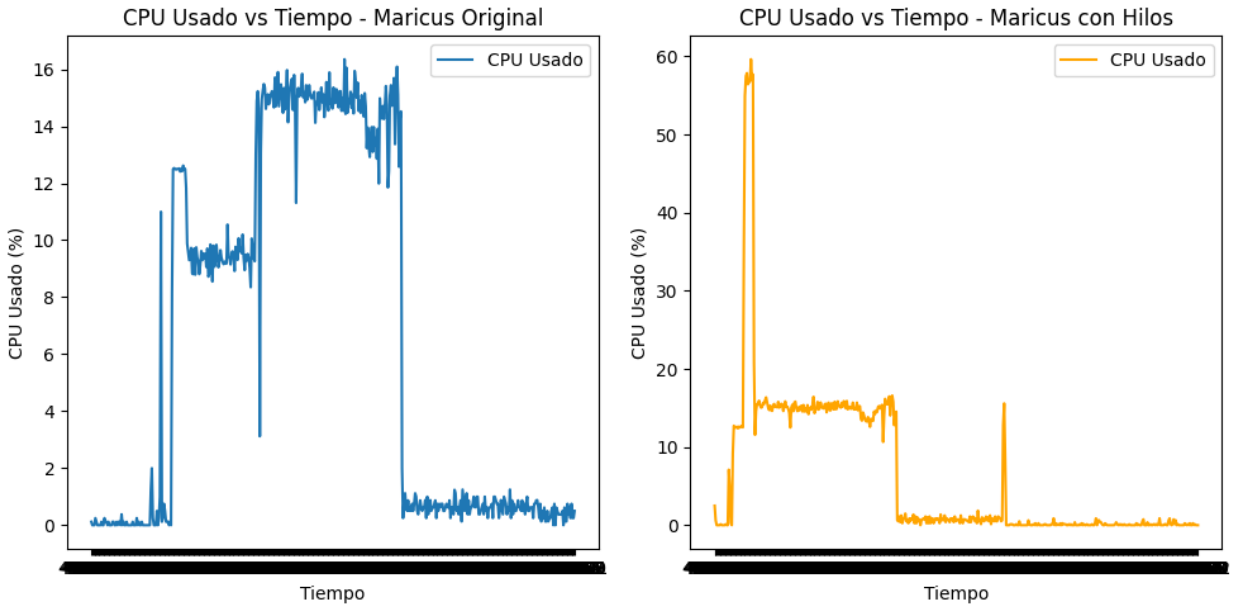
$$X = 0.084$$

Promedio de CPU Usado para Maricus con Hilos: 8.43%

En la Figura 25 se puede ver más a detalle el comportamiento del CPU usado en relación al tiempo, se ponen en comparativa las medidas tomadas en la ejecución tanto del programa original como del programa con hilos, para así ver la diferencia que existe entre ambas de manera más gráfica.

Figura 25

Uso de CPU para la ejecución de Maricus



Nota. La imagen muestra el uso del procesador (CPU) en la ejecución tanto de Maricus Original como con Hilos. La imagen se realizó en Colab, autoría propia.

Utilización de Memoria. En relación con la gestión de la memoria, se ha llevado a cabo una evaluación centrada en la medición de la utilización de la memoria RAM durante la ejecución del programa Maricus. Con el propósito de recopilar información detallada sobre el comportamiento de la memoria, se implementó un script especializado que registró datos relevantes durante todo el ciclo de vida del programa. Los resultados obtenidos indican que, en términos generales, la utilización de la memoria RAM se mantiene en un rango constante, que oscila entre 1.5 GB durante la ejecución del programa original de Maricus. Este análisis refleja la eficiencia de la gestión de la memoria, con un uso consistente y controlado de los recursos disponibles.

$$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$$

$$X = 0.0250$$

Promedio de RAM Usada para Maricus Original: 1.53 GB

Posteriormente, con la implementación del programa optimizado de Maricus, se realizó una nueva medición de la utilización de la memoria RAM, que usa el mismo script para la recolección de datos. Los resultados revelaron un cambio esperado en la utilización de la memoria que muestra un aumento moderado pero significativo.

$$X = \sum_{i=1an} (\frac{Ai}{Bi})/n$$

$$X = 0.0254$$

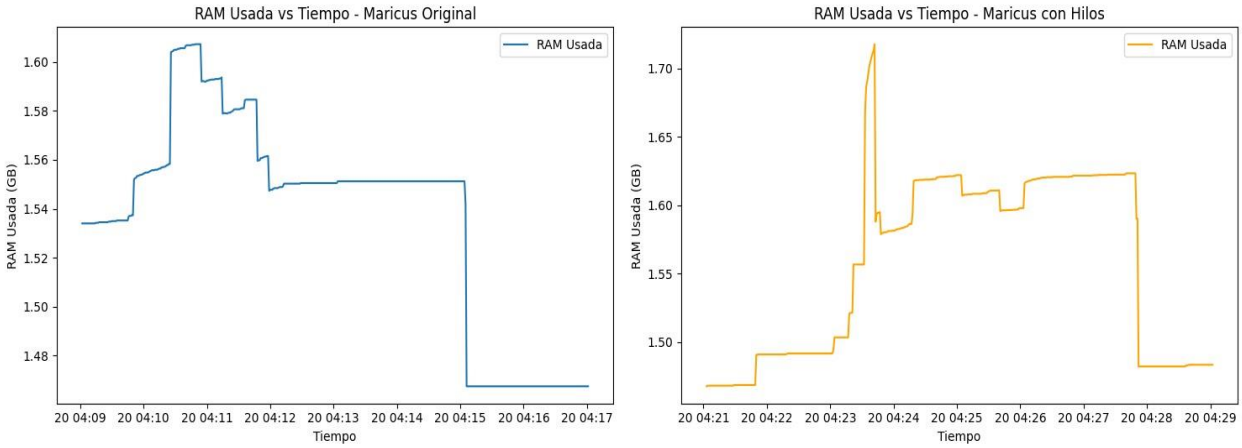
Promedio de RAM Usada para Maricus con Hilos: 1.57 GB

En la Figura 26 se puede ver más a detalle el comportamiento de la memoria usada en relación al tiempo, se ponen en comparativa las medidas tomadas en la ejecución tanto del programa

original como del programa con hilos, para así ver la diferencia que existe entre ambas de manera más gráfica.

Figura 26

Uso de RAM en la ejecución de Maricus



Nota. La imagen muestra el uso de la memoria RAM en la ejecución tanto de Maricus Original como con Hilos. La imagen se realizó en Colab, autoría propia.

En conclusión, la optimización del programa Maricus ha demostrado un impacto positivo en la utilización de la memoria RAM, con un aumento controlado que sugiere una gestión más eficiente de los recursos de memoria disponibles. Estos hallazgos respaldan la idea de que las modificaciones realizadas han conducido a una mejora significativa en la eficiencia del sistema en términos de gestión de recursos de memoria.

Madurez.

Eliminación de errores y Tiempo medio entre fallos. En la métrica de madurez, en particular para el tiempo medio entre fallos, se ha considerado como punto de referencia el tiempo de espera actual de Maricus al iniciarse. Dado que el programa cierra las conexiones y espera solo 52 segundos para recibir una respuesta de la base de datos, este período se ha identificado como el fallo más relevante. La rapidez con la que Maricus finaliza en ausencia de respuesta puede afectar la estabilidad del sistema y la capacidad de recuperación ante posibles interrupciones en la comunicación con la base de datos, además, como en la Figura 27 se puede observar otro tipo de falla relacionada con la lógica. Este análisis del tiempo medio entre fallos proporciona conceptos cruciales para evaluar la robustez del programa y permite identificar áreas clave que podrían beneficiarse de ajustes para mejorar la madurez y la fiabilidad del sistema en su conjunto.

Eliminación de Errores

$$X = \frac{Z}{N}$$

$$X = \frac{0}{2} \rightarrow X = 0$$

Tiempo medio entre fallos

$$X = \frac{T}{Z}$$

$$X = \frac{7{:}24\ min}{2}$$

Al aplicar la misma métrica al Maricus optimizado, es importante señalar que en este caso no se observa un tiempo medio entre fallos de manera convencional. En el programa optimizado, al no recibir respuesta en un plazo determinado, el programa se detiene automáticamente. Esta característica del programa optimizado elimina la noción de un tiempo

medio entre fallos tradicional, ya que el sistema está diseñado para reaccionar de manera inmediata ante la ausencia de respuesta, busca maximizar la eficiencia y garantizar una gestión rápida de las situaciones problemáticas. Este cambio en la dinámica de tiempo entre fallos refleja una estrategia proactiva para manejar posibles interrupciones, esto contribuye a la robustez y a la capacidad de recuperación del Maricus optimizado.

Eliminación de Errores

$$X = \frac{Z}{N}$$

$$X = \frac{1}{2} \rightarrow X = 0.5$$

Tiempo medio entre fallos

$$X = \frac{T}{Z}$$

$$X = \frac{4{:}39\ min}{1}$$

Figura 27

Errores presentes en el paso 5 de Maricus original

```
- 5. Insertar o actualizar boletos bsp y Capturador.
- - Numero de registro Insertar o actualizar: 9079
- (1406, "Data too long for column 'oidemision' at row 1")
- (1406, "Data too long for column 'oidemision' at row 1")
- (1406, "Data too long for column 'oidemision' at row 1")
- (1406, "Data too long for column 'oidemision' at row 1")
```

Nota. La imagen un error en el quinto paso de Maricus original, asociado a la columna de oidemision, autoría propia.

Disponibilidad.

Tiempo de actividad del sistema. En relación a la disponibilidad y el tiempo de actividad del sistema, se retoma lo mencionado en la situación inicial de Maricus, es crucial recordar que el programa cuenta con una ejecución automática programada a las 5 AM. Este enfoque busca aprovechar la disponibilidad del servidor en horas de baja actividad. Además, el tiempo en que el sistema está activo es proporcional al número de registros (meses) que se envían a procesar. Este aspecto significa que la disponibilidad del sistema está directamente vinculada a las solicitudes de procesamiento, adaptándose a la demanda y asegurar un uso eficiente de los recursos del servidor. Evaluar la disponibilidad en función de esta configuración permite comprender cómo Maricus gestiona las operaciones para maximizar su tiempo de actividad y cumplir con las solicitudes de procesamiento de manera efectiva. El tiempo del servicio de Maricus que está establecido en el cronograma operacional de la empresa, como se mencionó anteriormente se ejecuta con una tarea programada una vez al día, sin embargo, el tiempo del mismo es variante ya que depende de las fechas de entrada y salida que se procesen, como se puede ver en la Figura 28 se pone en comparativa los tiempos totales de la ejecución de ambas versiones de Maricus.

$$X = \frac{Z}{N}$$

$$X = \frac{7{:}24\ min}{7{:}24\ min} \rightarrow X = 1$$

En el caso del Maricus optimizado, también se ha evaluado el tiempo de actividad del sistema. Aunque la ejecución programada a las 5 AM se mantiene, se observa una reducción en el tiempo de actividad debido a la implementación de hilos. La introducción de hilos en la optimización ha permitido una mejora significativa en el tiempo de procesamiento Extraer, transformar y cargar (ETL) realizado por Maricus. Esta eficiencia derivada de la paralelización

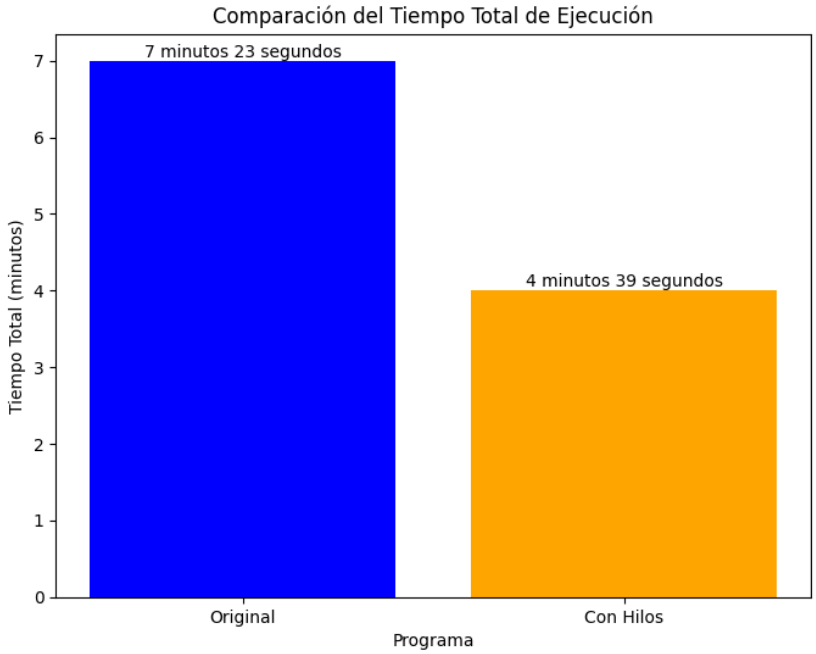
de tareas contribuye a una reducción en el tiempo de actividad del sistema, ofrece un equilibrio entre disponibilidad y eficiencia en la ejecución de procesos, y subrayar así los beneficios de las optimizaciones implementadas en el Maricus optimizado.

$$X = \frac{Z}{N}$$

$$X = \frac{4{:}39\ min}{4{:}39\ min} \rightarrow X = 1$$

Figura 28

Tiempo de actividad de Maricus Original y Maricus con hilos



Nota. La imagen muestra una comparativa temporal de la ejecución de Maricus Original con Maricus con hilos. La imagen se realizó con ayuda de Google Colab, autoría propia.

Tiempo de Inactividad del Sistema. En cuanto a la otra métrica de disponibilidad, la relativa al tiempo promedio de inactividad del sistema que se observa en la Figura 29, esta se refiere al periodo que Maricus permanece inactivo luego que ocurre un fallo o erro como se puede ver en la Figura 30. Esta métrica ofrece un panorama sobre los períodos de inactividad no planificados del sistema, en cuanto a Maricus original se encuentra dos errores asociados a la conexión y la lógica dentro de la lógica en la etapa de transformación de la data.

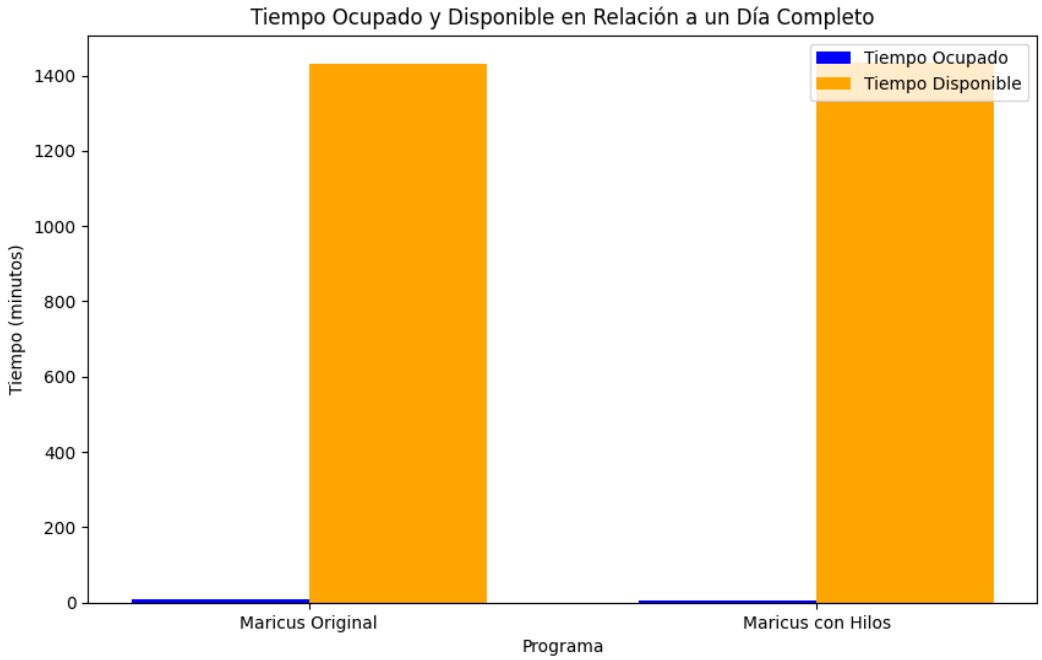
$$X = \frac{2}{53\ seg}$$

Mientras que en Maricus con hilos solo se enfrenta a una falla asociada a la transformación de los datos, que tarda alrededor de un segundo, en pasar de ella. Esos datos son representados en la siguiente fórmula:

$$X = \frac{1}{1\ seg}$$

Figura 29

Tiempo de disponibilidad del programa Maricus



Nota. La imagen muestra el tiempo disponible que existe tras la ejecución de Maricus Original y Maricus con Hilos. La imagen fue realizada con Google Colab, autoría propia.

Tolerancia a Fallos.

Anulación de Operación Incorrecta. En el contexto de la métrica de tolerancia a fallos, especialmente en lo que respecta a la anulación de operaciones incorrectas a lo largo del programa, es crucial destacar que Maricus, tanto en su versión original como en la optimizada, cuenta con funciones específicas para corregir errores identificados en los datos procesados. Por ejemplo, si se encuentra un boleto con un canal no identificado, el programa incorpora funciones diseñadas para abordar y corregir este tipo de inconvenientes. Esta capacidad de autocorrección contribuye significativamente a la tolerancia a fallos del sistema, que asegura que, en caso de identificar datos incorrectos o inconsistentes, Maricus pueda abordarlos de manera proactiva y continuar con su ejecución sin afectar la integridad de los resultados finales. La consistencia en esta capacidad entre la versión original y optimizada destaca la importancia dada a la tolerancia a fallos a lo largo del desarrollo y mejora del programa, es así que, se identificó dos fallas en el programa de las cuales ninguna de ellas fue corregida en el Maricus Original.

$$X = No\ aplica$$

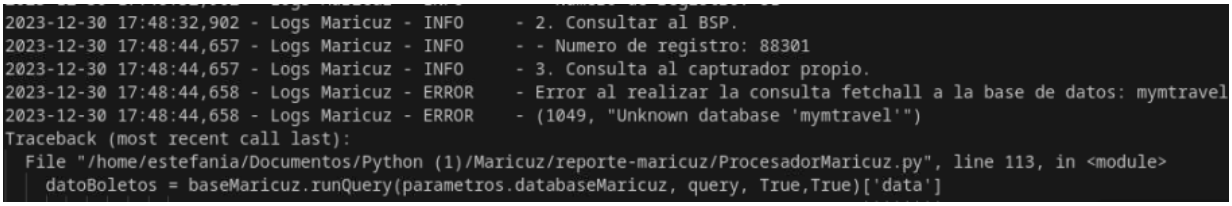
En la versión con Hilos de Maricus se identificó los errores antes mencionados, y se implementó un bloque de código en la clase conexión que permitió solventar una de las fallas al momento de recuperar información desde ella.

$$X = \frac{1}{2}$$

$$X = 0.5$$

Figura 30

Error con la conexión a la Base de Datos del Capturador propio



Nota. La imagen comprende el error en ejecución de Maricus original del paso tres, en el que se hace una consulta a la base de datos de Capturador propio. La imagen es una captura de un entorno de ejecución, autoría propia.

Cálculo máximo de hilos

En el proceso de calcular el número óptimo de hilos, se ha tenido en cuenta el límite inherente de la biblioteca de Python que se usa, específicamente `concurrent.futures`. Para determinar la cantidad adecuada de hilos, el programa se sometió a pruebas con ayuda de la función que se presenta en la Figura 31. Dicha función fue diseñada para evaluar el rendimiento del programa con diferentes configuraciones de hilos. Las pruebas con distintos números de hilos han proporcionado valiosa información sobre la escalabilidad del programa en función de la cantidad de recursos paralelos disponibles.

Figura 31

Código que determina el número de Hilos

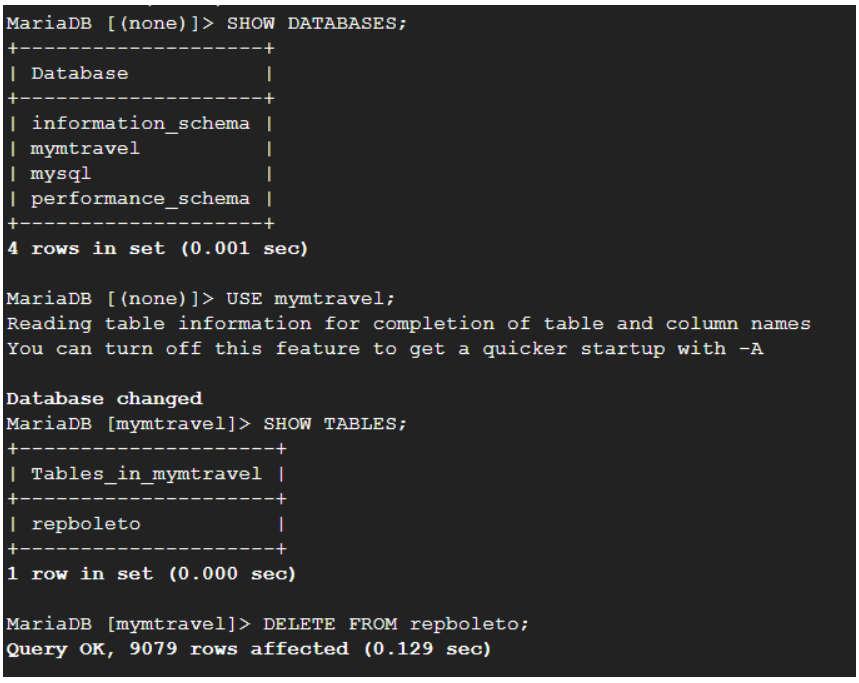
```
def generarHiloParametrosConRetorno(self):
    results = []
    n = psutil.cpu_count(logical=True)
```

Nota. La imagen muestra la función proporcionada por la biblioteca psutil que devuelve el número lógico de núcleos de CPU disponibles en el sistema. La imagen es una captura de un entorno de ejecución, autoría propia.

Además, durante el proceso de determinar la cantidad óptima de hilos, se llevó a cabo una evaluación sistemática que usa incrementos de cinco hilos en cada prueba. Cada ejecución de prueba incluyó un paso esencial para garantizar la precisión y completitud del proceso ETL: la limpieza de la información en la tabla repboleto como se ve en la Figura 32. En cada iteración, la data de esta tabla se vació antes de ejecutar el programa con la configuración específica de hilos. Esta práctica aseguró que cada prueba se realizara en condiciones consistentes, para así evitar cualquier interferencia de datos previos y permite una evaluación precisa del rendimiento del programa con la variación en el número de hilos.

Figura 32

Base de datos usada en la ejecución de Maricus

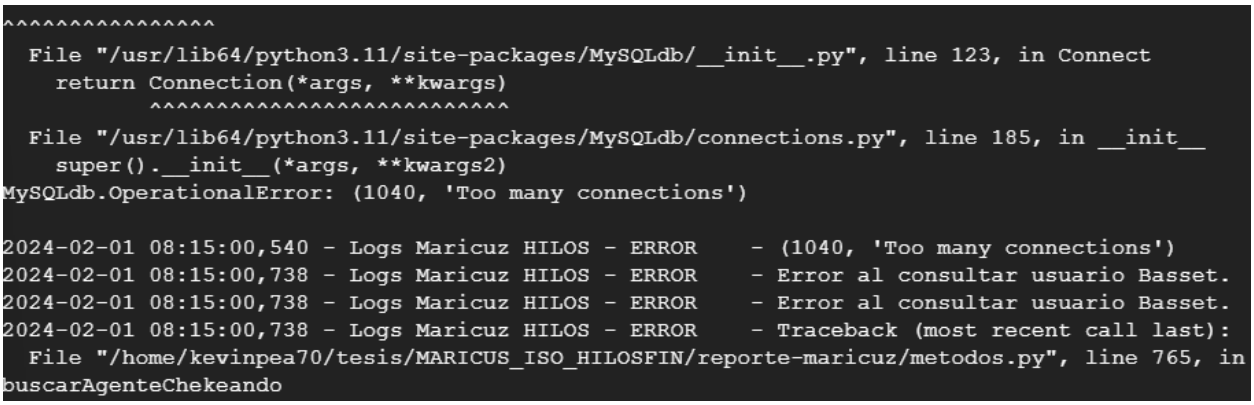


Nota. La imagen muestra la base de datos que contiene la data después del ETL. La imagen es una captura de un entorno de ejecución, autoría propia.

No obstante, al llevar a cabo pruebas con un número más elevado de hilos, específicamente 150 hilos, se identificó un inconveniente crítico. El programa arrojó un error como se puede ver en la Figura 33, debido a que la base de datos no admitía un número tan elevado de conexiones concurrentes. Este contratiempo subraya la importancia de considerar las limitaciones de los recursos subyacentes, como la capacidad de la base de datos, al determinar el número óptimo de hilos. Este incidente proporciona una valiosa perspectiva sobre los límites prácticos del sistema y destaca la necesidad de equilibrar la concurrencia con la capacidad de los recursos disponibles, los resultados de esa prueba se pueden ver a detalle en la Figura 34.

Figura 33

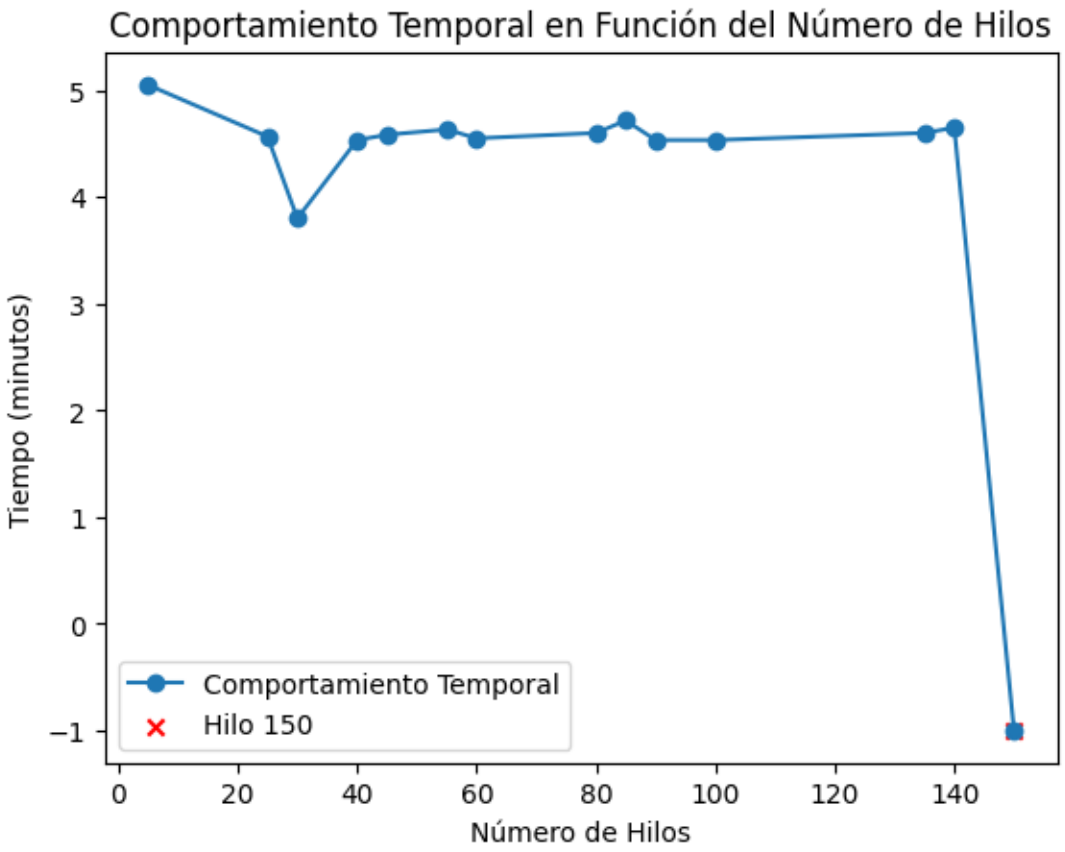
Número máximo de hilos



Nota. Se muestra el error producido al ejecutar Maricus con 50 hilos. La imagen es una captura de un entorno de ejecución, autoría propia.

Figura 34

Comportamiento temporal de Maricus



Nota. Se presenta el comportamiento temporal de Maricus en función del número de hilos que fueron probados. La imagen es una captura de un entorno de ejecución, autoría propia.

Una vez determinado el número de hilos por tiempo, para evitar tener un valor atípico en la lectura de los datos se volvió a realizar la misma prueba, y así encontrar similitud entre las dos mediciones, sin embargo, los valores de la nueva medición resultaron ser totalmente diferentes a los de la primera, lo que sugiere que los valores son atípicos o que no existe relación evidente entre el número de hilos y el tiempo.

Tabla 15

Nueva toma de mediciones para los hilos

Hilos	Hora_Inicio	Hora_Fin	Tiempo
20	1:49:31	1:54:50	0:05:19
25	1:57:52	2:02:45	0:04:53
30	2:05:44	2:10:42	0:04:58
30	2:14:58	2:19:57	0:04:59
35	2:28:27	2:33:19	0:04:52
65	2:41:01	2:46:00	0:04:59

Nota. Se presenta el comportamiento temporal de Maricus en función del número de hilos de la segunda muestra. La imagen es una captura de un entorno de ejecución, autoría propia.

Como se observa en la Tabla 15, al volver a comprobar el valor óptimo de la Figura 34, ya no es 30 hilos a su vez en La Figura 10, se muestra que el servidor tiene dos hilos por cuatro cores, por lo que la reducción de tiempo por hilos depende únicamente del número de hilos por core del procesador.

Estas pruebas externas arrojaron resultados que revelaron áreas de mejora, especialmente en el motor de almacenamiento de la base de datos "Repoboleto", que inicialmente utilizaba InnoDB y presentaba ciertas restricciones al emplear hilos en las peticiones a las bases de datos. Para optimizar la eficiencia del programa en este aspecto, se tomó la decisión de cambiar el motor de almacenamiento a MyISAM, con lo cual se aprovecha sus beneficios particulares, como el tipo de bloqueo que ofrece. Este cambio se realizó con el propósito de superar las restricciones y limitaciones asociadas a InnoDB al utilizar hilos. Tras la implementación de este ajuste, se verificó su efectividad, por lo cual se confirma que el programa funcionó de manera más eficiente y respondió de manera más efectiva a las operaciones concurrentes.

Las configuraciones presentes en la base de datos que se toman en cuenta en este caso, son las siguientes, cuyo valor se muestra en paréntesis seguido de la definición:

- aria_repair_threads (1): Número de hilos utilizados para la reparación automática de tablas Aria.
- binlog_optimize_thread_scheduling (ON): Controla si MySQL intenta optimizar la programación de hilos para la replicación binlog.
- debug_no_thread_alarm (OFF): Habilita o deshabilita las alarmas de hilo para depuración.
- innodb_encryption_threads (0): Número de hilos utilizados para la encriptación InnoDB.
- innodb_purge_threads (4): Número de hilos utilizados para el proceso de purga en InnoDB.
- innodb_read_io_threads (4): Número de hilos utilizados para operaciones de lectura de E/S en InnoDB.
- innodb_thread_concurrency (0): Número máximo de hilos concurrentes en InnoDB.
- innodb_thread_sleep_delay (0): Retraso en microsegundos que los hilos de InnoDB duermen antes de volver a intentar operaciones de I/O.
- innodb_write_io_threads (4): Número de hilos utilizados para operaciones de escritura de E/S en InnoDB.
- max_delayed_threads (20): Número máximo de hilos para la inserción diferida máxima.
- max_insert_delayed_threads (20): Número máximo de hilos para inserciones diferidas.
- myisam_repair_threads (1): Número de hilos utilizados para la reparación automática de tablas MyISAM.
- performance_schema_max_thread_classes (50): Número máximo de clases de hilos en el esquema de rendimiento.

- performance_schema_max_thread_instances (-1): Número máximo de instancias de hilos en el esquema de rendimiento.
- pseudo_thread_id (8): ID de hilo pseudoasignado.
- slave_domain_parallel_threads (0): Número de hilos paralelos utilizados para replicación de dominio.
- slave_parallel_threads (0): Número de hilos paralelos utilizados para replicación de esclavos.
- thread_cache_size (151): Tamaño del caché de hilos.
- thread_handling (one-thread-per-connection): Manejo de hilos por conexión.
- thread_pool_dedicated_listener (OFF): Controla si el hilo de escucha dedicado para el grupo de subprocesos de MySQL está habilitado.
- thread_pool_exact_stats (OFF): Controla si las estadísticas exactas del grupo de subprocesos de MySQL están habilitadas.
- thread_pool_idle_timeout (60): Tiempo de espera en segundos antes de que los subprocesos inactivos se terminen en el grupo de subprocesos de MySQL.
- thread_pool_max_threads (65536): Número máximo de subprocesos en el grupo de subprocesos de MySQL.
- thread_pool_oversubscribe (3): Factor de sobreasignación de subprocesos en el grupo de subprocesos de MySQL.
- thread_pool_prio_kickup_timer (1000): Intervalo de tiempo en milisegundos para aumentar la prioridad del hilo en el grupo de subprocesos de MySQL.
- thread_pool_priority (auto): Prioridad de subprocesos en el grupo de subprocesos de MySQL.
- thread_pool_size (8): Tamaño inicial del grupo de subprocesos de MySQL.

- thread_pool_stall_limit (500): Límite en milisegundos para la detección de bloqueos en el grupo de subprocesos de MySQL.
- thread_stack (299008): Tamaño de la pila de hilos en bytes.
- wsrep_slave_threads (1): Número de hilos esclavos utilizados por WSREP.

En el proceso de desarrollo, se implementaron hilos en determinados pasos de ejecución, lo que permite a Maricus aprovechar el paralelismo y mejorar la eficiencia del programa. Además, como parte de la optimización para facilitar la concurrencia de las solicitudes, se llevó a cabo un cambio en el archivo de conexiones. Este ajuste específico en el archivo de conexiones ha sido diseñado para mejorar la gestión de las peticiones y su ejecución en paralelo, así contribuye a una mayor eficacia y rendimiento del programa en entornos multiproceso. Estas mejoras se han realizado con el objetivo de maximizar el aprovechamiento de recursos y mejorar la velocidad de ejecución del sistema.

Capítulo IV

En este capítulo, se analizan los diferentes resultados obtenidos de acuerdo a la obtención de las métricas implementadas en Maricus original como en Maricus con hilos, aplicadas al caso de estudio en consideración se puede evaluar los siguientes resultados, que se presentan en las tablas 22 y 23 con las ponderaciones en las subcaracterísticas para la eficiencia de desempeño y la fiabilidad.

Análisis del Producto Software

El producto software como objeto de estudio de comparación entre un ambiente normal a uno en el que se implementó hilos, es el programa Maricus, el cual, como se mencionó anteriormente es crucial en la elaboración de Reportes dentro de la empresa.

Principales Componentes

El principal componente de software para el objeto del estudio, ha sido considerado, el programa Maricus en su totalidad como se describe en la Tabla 16, de forma más detallada.

Tabla 16

Principales Componentes del Software

N.	Nombre	Contexto	Descripción
1	Maricus	ETL integrado	Maricus es un componente de software de backend diseñado para la ejecución eficiente de procesos Extraer, transformar y cargar (ETL) en entornos basados en Python. Su enfoque se centra en la simplicidad y la automatización de la manipulación de datos, esto permite a los usuarios realizar tareas ETL de manera rápida y efectiva.

Nota. Componentes de software a ser evaluados. Autoría propia.

Herramientas de Evaluación de Código Fuente

Las herramientas seleccionadas son de software libre y su utilización está disponible al público, a continuación, en la Tabla 17, se detallan sus características.

Tabla 17

Herramientas de Evaluación de Código

N.	Nombre	Descripción	Sitio Web
1	Google Colab	Plataforma de colaboración en línea	https://colab.research.google.com/

Nota. Herramientas disponibles para la calidad del código fuente. Autoría propia.

Evaluación del Producto Software

Durante el proceso de evaluación, se llevó a cabo un exhaustivo análisis comparativo entre el programa Maricus ejecutado con Hilos y la versión original de Maricus. Este análisis se basó en las directrices y metodologías detalladas en los capítulos 2 y 3, centrándose especialmente en las métricas asociadas a las subcaracterísticas establecidas por el enfoque de eficiencia de desempeño y fiabilidad de la norma ISO 25010.

Componentes de la calidad

Las tablas 23 y 24 contienen los parámetros con los que se consignan las métricas de calidad, a partir de las cuales se puede dar valor a cada subcaracterística definida con anterioridad en los componentes de eficiencia de desempeño y fiabilidad que componen parte esencial de esta evaluación.

Ambos componentes hacen uso de los siguientes campos:

- Característica: hace referencia al nombre de la característica parte de la ISO que se va a evaluar.
- Subcaracterística: nombre de la subcaracterística que se evalúa.
- Métrica: elemento específico de una subcaracterística a evaluar.
- Fórmula: presenta la fórmula y variables de la misma, a usar en la evaluación de una métrica.
- Valor deseado: representa el valor máximo a ser obtenido luego de aplicar la fórmula respectiva.
- Aplica: en este se pone si la métrica va a ser aplicada o no.
- Valor obtenido: equivale al valor de X que se obtuvo tras aplicar la fórmula.
- Ponderación /10: se refiere al peso o importancia asignada a una característica

específica de calidad.

- Valor parcial total /10 : este valor se obtiene al multiplicar la ponderación asignada a una característica específica por la puntuación alcanzada en esa característica
- Nivel de importancia: es una indicación cualitativa de la relevancia de la característica evaluada que puede ser Alto(A), Medio(M) y Bajo(B).
- Porcentaje de importancia: representa la contribución de una característica específica al logro del objetivo general de calidad del software
- Porcentaje final: refleja la contribución general de todas las características evaluadas al logro del objetivo global de calidad del software.
- Calidad del sistema /10: describe el valor final de las características de calidad aplicadas.

Ponderación de Porcentajes de las características

Tabla 18

Ponderación de las características de la ISO 25010

Subcaracterística	Nivel de Importancia	Ponderación	Modo de ponderación
Eficiencia de desempeño	A	60%	La asignación del 60% refleja la importancia crítica del tiempo y usos de recurso.
Fiabilidad	C	40%	La asignación del 40% refleja la importancia del funcionamiento correcto del programa.

Nota. Ponderación de las características de la ISO 25010. Autoría propia.

Ponderación de Porcentajes

En el siguiente apartado se muestran los porcentajes correspondientes a las ponderaciones para cada componente de la ISO 25010, así como su subcaracterísticas. Se describe más a detalle en las Tablas 19 y 20.

Tabla 19

Ponderación en las Subcaracterísticas de Eficiencia de Desempeño

Subcaracterística	Nivel de Importancia	Ponderación	Modo de ponderación
Comportamiento temporal	A	80%	La asignación del 80% refleja la importancia crítica del tiempo en la eficiencia de este proceso.
Utilización de recursos	C	20%	La asignación del 20% refleja la importancia de optimizar el consumo de recursos, lo que contribuye a un rendimiento general eficiente y evita posibles cuellos de botella en el sistema.

Nota. Valores a ser ponderados en las métricas seleccionadas. Adaptado de Rivera Guaraca, M. I. (2021). Evaluación de la fiabilidad en el sistema web de agendamiento de citas médicas del Hospital General Universitario Andino de la Provincia de Chimborazo (Bachelor's thesis, Riobamba, Universidad Nacional de Chimborazo).

Tabla 20

Ponderación en las Subcaracterísticas de Fiabilidad

Subcaracterística	Nivel de Importancia	Ponderación	Modo de ponderación
Madurez	C	30%	En el contexto del ETL, un programa maduro es crucial para garantizar que Maricus pueda funcionar de manera estable y confiable durante operaciones prolongadas. La asignación del 30% refleja la importancia de la estabilidad a lo largo del tiempo.
Disponibilidad	C	20%	En el caso del ETL, la disponibilidad constante es esencial para garantizar que el proceso pueda ejecutarse según el cronograma deseado. La asignación del 20% refleja la importancia de la disponibilidad continua.
Tolerancia a fallos	B	50%	En el ETL, la capacidad de continuar o recuperarse de manera efectiva después de un fallo es crucial para garantizar la fiabilidad del proceso. La asignación del 50% refleja la importancia de la resiliencia frente a fallos.

Nota. Valores a ser ponderados en las métricas seleccionadas. Adaptado de Rivera Guaraca, M. I. (2021). Evaluación de la fiabilidad en el sistema web de agendamiento de citas médicas del

Hospital General Universitario Andino de la Provincia de Chimborazo (Bachelor's thesis, Riobamba, Universidad Nacional de Chimborazo).

Aplicación de Métricas

A continuación, en la Tabla 21 y 22, se presentan el resultado de las métricas seleccionadas para cada una de las versiones de Maricus, así como la evaluación de las métricas que intervienen en el proceso.

Tabla 21

Métricas Aplicadas a Maricus Original y Maricus con Hilos para Eficiencia de Desempeño

N.	Métrica	Resultados Maricus Original	Resultados Maricus con Hilos
1	Tiempo de respuesta	18 seg	13 seg
2	Tiempo de espera	7 min 24 seg	4 min 39 seg
3	Rendimiento	24/7:24 min	24/ 4:39 min
4	Utilización de CPU	6.41% - 0.073	8.43%- 0.084
5	Utilización de memoria	1.53 GB -0.0250	1.55 GB – 0.0254

Nota. Resultados de la evaluación basada en los parámetros de la ISO 25010 para eficiencia de desempeño. Autoría propia.

Tabla 22

Métricas Aplicadas a Maricus Original y Maricus con Hilos para Fiabilidad

N.	Métrica	Resultados Maricus Original	Resultados Maricus con Hilos
1	Tiempo medio entre fallos	7:24min/2	4:39min/1
2	Eliminación de errores	0	0.50
3	Tiempo de actividad del sistema	1	1

N.	Métrica	Resultados Maricus Original	Resultados Maricus con Hilos
4	Tiempo promedio de inactividad del sistema	2/52seg	1/1seg
5	Anulación de la operación incorrecta	No aplica	0.5

Nota. Resultados de la evaluación basada en los parámetros de la ISO 25010 para la Fiabilidad. Autoría propia.

Tabla 23

Métricas Aplicadas a Maricus Original

CARACTERÍSTICA	SUBCARACTERÍSTICA	MÉTRICA	FÓRMULA	VALOR DESEADO	APLICA	VALOR OBTENIDO	PONDERACIÓN /10	VALOR PARCIAL TOTAL /10	NIVEL DE IMPORTANCIA	PORCENTAJE DE IMPORTANCIA	PORCENTAJE FINAL	CALIDAD DEL SISTEMA /10
Eficiencia de Desempeño	Comportamiento temporal	Tiempo de respuesta	X = Z-N N=Tiempo de envío de la petición Z=Tiempo de espera para recibir la primera respuesta	0<=x<=10s El peor caso x > 10 s	Sí	18 s	5.55	5.45	A	80% (4.36 %)	60% (3.66%)	**3.73**
		Tiempo de espera	X=Z-N N=Tiempo inicial de una tarea Z=Tiempo final de la tarea	0<=x<=4min Mientras más cerca del 0 es mejor. Este el peor caso >=4 min	Sí	7 min 24 seg 444 s	5.40					
		Rendimiento	X=N/W N=Tareas completadas W=Tiempo	El mejor caso 0.1 El peor caso 0/6	Sí	24/ 7 min 24 seg 0.054	5.4					
	Utilización de recursos	Utilización de CPU	X = Z Z=Cantidad de CPU que es usada para completar una tarea	Deseado: 0.08 Este el peor caso 0.1	Sí	6.41% - 0.073	9.12	8.72	C	20% (1.744 %)		
		Utilización de memoria	X = Z Z= Cantidad de memoria que es usada para completar una tarea	Deseado: 0.03 Este el peor caso 0.1	Sí	1.53 GB - 0.0250	8.33					
Fiabilidad	Madurez	Tiempo medio entre fallos	X =T/Z Z= Número total de fallas detectadas actualmente T= Tiempo de operación T>0	Deseado: 60seg Este el peor caso 300seg	Sí	7 min 24 seg/2 222 seg	2.70	1.35	C	30% (0.405%)	40% (0.067 %)	
		Eliminación de errores	X=Z/N Z= Número de fallas corregidas en la fase de diseño/codificación /pruebas N = Número de fallas detectadas en las pruebas N>0	Deseado: 1 Este el peor caso 0	Sí	0	0					

CARACTERÍSTICA	SUBCARACTERÍSTICA	MÉTRICA	FÓRMULA	VALOR DESEADO	APLICA	VALOR OBTENIDO	PONDERACIÓN /10	VALOR PARCIAL TOTAL /10	NIVEL DE IMPORTANCIA	PORCENTAJE DE IMPORTANCIA	PORCENTAJE FINAL	CALIDAD DEL SISTEMA /10
	Disponibilidad	Tiempo de actividad del sistema	X = Z/N Z=Tiempo del servicio proporcionado actualmente N=Tiempo de servicio del sistema regulado en el cronograma operacional N>0	Deseado: 1 Este el peor caso ≠1	Sí	1	10	6.32	C	20% (1.264 %)		
		Tiempo promedio de inactividad del sistema	X=T/Z Z=Número total de fallas detectadas actualmente T=Tiempo de la inactividad T>0	Deseado: 1 Este el peor caso > 600 seg	Sí	53/2 seg 26.55	2.65					
	Tolerancia a fallos	Anulación de la operación incorrecta	X = Z/N Z=Número de operaciones incorrectas encontradas N=Número de funciones implementadas para anular las operaciones incorrectas N>0	Deseado: 1 Este el peor caso 0	No	-	-	0	B	50% (0%)		

Nota. Resultados de la aplicación de las métricas a Maricus Original. Adaptado de ISO 25010. (n.d.).

https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&start=6

Tabla 24

Métricas Aplicadas a Maricus con hilos

SUBCARACTERÍSTICA	MÉTRICA	FÓRMULA	VALOR DESEADO	APLICA	VALOR OBTENIDO	PONDERACIÓN /10	VALOR PARCIAL TOTAL /10	NIVEL DE IMPORTANCIA	PORCENTAJE DE IMPORTANCIA	PORCENTAJE FINAL	CALIDAD DEL SISTEMA /10
Comportamiento temporal	Tiempo de respuesta	X = Z-N N=Tiempo de envío de la petición Z=Tiempo de espera para recibir la primera respuesta	0<=x<=10 s	Sí	13 seg	7.69	8.29	A	80% (6.632 %)	60% (5.08)	**7.30**
	Tiempo de espera	X=Z-N N=Tiempo inicial de una tarea Z=Tiempo final de la tarea	0<=x<=4 min	Sí	4 min 39 seg 279 seg	8.60					
	Rendimiento	X=N/W N=Tareas completadas W=Tiempo	El mejor caso 0.1 El peor caso 0/6	Sí	24/4 min 39 seg 0.086	8.6					
Utilización de recursos	Utilización de CPU	X = Z Z=Cantidad de CPU que es usada para completar una tarea	Deseado: 0.08 Este el peor caso 0.1	Sí	8.43 % - 0.084	10	9.23	C	20% (1.842)		
	Utilización de memoria	X = Z Z= Cantidad de memoria que es usada para completar una tarea	Deseado: 0.03 Este el peor caso 0.1	Sí	1.55 GB - 0.0254	8.46					
Madurez	Tiempo medio entre fallos	X = T/Z Z= Número total de fallas detectadas actualmente T= Tiempo de operación T>0	Deseado: 60seg Este el peor caso 300 seg	Sí	4 min 39 seg/ 279 seg	2.15	3.58	C	30% (1.074 %)	40% (2.23)	

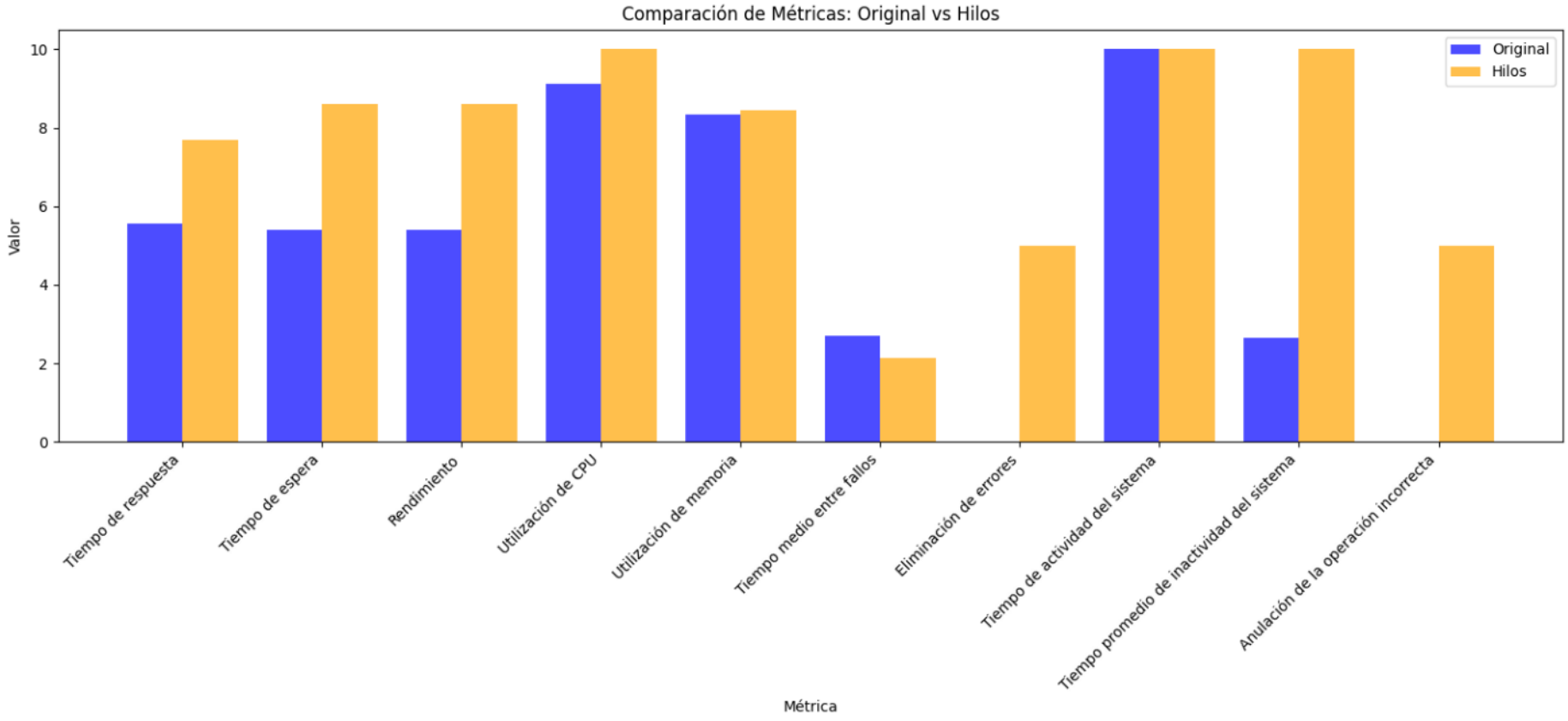
SUBCARACTERÍSTICA	MÉTRICA	FÓRMULA	VALOR DESEADO	APLICA	VALOR OBTENIDO	PONDERACIÓN /10	VALOR PARCIAL TOTAL /10	NIVEL DE IMPORTANCIA	PORCENTAJE DE IMPORTANCIA	PORCENTAJE FINAL	CALIDAD DEL SISTEMA /10
	Eliminación de errores	X=Z/N Z= Número de fallas corregidas en la fase de diseño/codificación /pruebas N = Número de fallas detectadas en las pruebas N>0	Deseado: 1 Este el peor caso 0	Sí	0.50	5					
Disponibilidad	Tiempo de actividad del sistema	X = Z/N Z=Tiempo del servicio proporcionado actualmente N=Tiempo de servicio del sistema regulado en el cronograma operacional N>0	0<=x<=1 Mientras más cerca del 1 es mejor	Sí	1	10	10	C	20% (2 %)		
	Tiempo promedio de inactividad del sistema	X=T/Z Z=Número total de fallas detectadas actualmente T=Tiempo de la inactividad T>0	Deseado: 1 Este el peor caso > 600 seg	Sí	1 seg	10					
Tolerancia a fallos	Anulación de la operación incorrecta	X = Z/N Z=Número de operaciones incorrectas encontradas N=Número de funciones implementadas para anular las operaciones incorrectas N>0	Deseado: 1 Este el peor caso 0	Sí	0.5	5	5	B	50% (2.5 %)		

Nota. Resultados de la aplicación de las métricas a Maricus con hilos. Adaptado de

ISO 25010. (n.d.). https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&start=6

Figura 35

Gráfico Comparativo entre Maricus Original y Maricus con hilos



Nota. El gráfico muestra el resultado del estudio entre las métricas en ambas versiones de Maricus. Autoría Propia.

Análisis de los Resultados

Una vez implementados los hilos en el proceso ETL de la empresa "Casa de Incentivos Casintour" durante el año 2023, al evaluar los resultados mediante el marco de referencia ISO 25010, se realizó una exhaustiva comparación con la versión original de Maricus que actualmente está en producción. Este análisis permitió obtener una visión más clara y detallada sobre si existió o no una mejora.

Valores Presentados

Se destaca la información contenida en las tablas 23 y 24, las cuales revelan que, en términos de métricas de eficiencia de desempeño, la implementación de hilos emerge como una opción favorable. Estos resultados son particularmente significativos para una empresa cuyo recurso valioso es el tiempo, ya que la capacidad de ofrecer informes de manera oportuna se traduce en una ventaja competitiva en el mercado.

El sistema tradicional obtuvo una puntuación de 3.73 que representa que aún existen áreas deficientes en las métricas de eficiencia de desempeño y fiabilidad.

Comportamiento Temporal.

Tiempo de Respuesta. Se obtuvo un valor de 18 segundos sobre un valor deseado de entre 0 y 10 segundos Lo que da una nota resultante de 5.55 sobre 10.

Tiempo de Espera. Sobre un valor deseado de entre 0 y 4 min, se obtuvo un valor de 7:24 min. Lo que da una nota resultante de 5.40 sobre 10.

Rendimiento. Al considerar el mejor caso 0.1 se obtuvo un valor de 0.054, cuya nota sobre 10 es un total de 5.4.

Utilización de Recursos.

Utilización de CPU. Con el mejor caso 0.08 se obtuvo un valor de 0.073 cuya nota sobre 10 es un total de 9.12.

Utilización de Memoria. Al considerar el mejor caso 0.03, se obtuvo un valor de 0.0250 cuya nota sobre 10 es un total de 8.33.

Madurez.

Tiempo Medio Entre Fallos. Con el valor deseado de 60 segundos, se obtuvo un valor de 7min y 24 seg, su nota sobre 10 es un total de 2.70.

Eliminación de Errores. Al considerar el mejor caso 1, se obtuvo un valor de 0, por lo tanto, la nota sobre 10 es 0.

Disponibilidad.

Tiempo de Actividad del Sistema. El valor deseado en este caso es 1, se obtuvo un valor de 1, por lo tanto, la nota sobre 10 es 10.

Tiempo Promedio de Inactividad del Sistema. Al considerar el mejor caso 1, se tiene un valor de 26.55 cuya nota sobre 10 es un total de 2.65.

Tolerancia a fallos.

Anulación de Operación Incorrecta. Está métrica no aplica, puesto que no se desarrolló nada sobre el sistema tradicional, para la solución de errores.

Mientras que en la versión actualizada se obtuvo una puntuación de 7.30, que evidencia la mejora significativa de varios aspectos, en las métricas:

Comportamiento Temporal.

Tiempo de Respuesta. Se obtuvo un valor de 13 seg sobre un valor deseado de entre 0 y 10 seg. Lo que da una nota resultante de 7.69 sobre 10.

Tiempo de Espera. Sobre un valor deseado de entre 0 y 4 min, se obtuvo un valor de 4:39 min. Lo que da una nota resultante de 8.60 sobre 10.

Rendimiento. Al considerar el mejor caso 0.1 se obtuvo un valor de 0.086, cuya nota sobre 10 es un total de 8.6.

Utilización de Recursos.

Utilización de CPU. Con el mejor caso 0.08 se obtuvo un valor de 0.084 cuya nota sobre 10 es un total de 10.

Utilización de Memoria. Al considerar el mejor caso 0.03, se obtuvo un valor de 0.0254 cuya nota sobre 10 es un total de 8.46.

Madurez.

Tiempo Medio Entre Fallos. Con el valor deseado de 60 segundos, se obtuvo un valor de 4min y 39 seg, su nota sobre 10 es un total de 2.15.

Eliminación de Errores. Al considerar el mejor caso 1, se obtuvo un valor de 0.5, por lo tanto, la nota sobre 10 es 5.

Disponibilidad.

Tiempo de Actividad del Sistema. El valor deseado en este caso es 1, se obtuvo un valor de 1, por lo tanto, la nota sobre 10 es 10.

Tiempo Promedio de Inactividad del Sistema. Al considerar el mejor caso 1, se tiene un valor de 1 cuya nota sobre 10 es un total de 10.

Tolerancia a Fallos.

Anulación de Operación Incorrecta. Al considerar el mejor caso 1, se tiene un valor de 0.5 cuya nota sobre 10 es un total de 5.

Evaluación Adicional

Es crucial destacar la importancia de considerar la naturaleza de Maricus como un programa consecutivo, que carece de módulos y que opera mediante pasos secuenciales. Esta característica fundamental influye en la metodología de evaluación, ya que Maricus se percibe como un único programa ejecutado de manera secuencial. Al contrastar esto con la evaluación de métricas ISO, se observa que la dinámica de un programa consecutivo difiere significativamente de la evaluación de una aplicación web, donde la interconexión de módulos y procesos concurrentes puede influir en la interpretación de las métricas. Esta consideración adicional resalta la necesidad de abordar la singularidad de Maricus en el proceso de evaluación y destaca la importancia de adaptar las métricas y enfoques de evaluación según la naturaleza específica del programa bajo análisis.

Capítulo V

Conclusión y Recomendaciones

En conclusión, se cumplió con el objetivo general de proponer una estrategia de implementación que aproveche el paralelismo basado en procesos para mejorar la eficiencia y el rendimiento del ETL en Big Data de la empresa “Casa de Incentivos Casintour" durante el año 2023, para ello se implementó una clase denominada hilos.py

con ayuda del patrón de diseño Factory, el cual gestiona varias peticiones a partir de bloques de hilos.

La evaluación de los resultados se hizo con la ayuda del marco ISO 25010, lo que proporciona una visión clara y detallada de los beneficios potenciales, especialmente en términos de ahorro de tiempo con un puntaje resultante de 7.30 que fue producto de la suma del puntaje que se obtuvo en eficiencia de desempeño con 5.08 y fiabilidad de 2.23, por otra parte, el programa original obtuvo un puntaje global de 3.73 que fue resultado de la suma del puntaje que se obtuvo en eficiencia de desempeño con 3.66 y fiabilidad de 0.067

De igual manera se alcanzó a comprender los conceptos y principios fundamentales del proceso ETL en entornos de Big Data, para ello se aplicaron las técnicas paralelismo basado en proceso mediante el uso de librerías de Python centradas en la manipulación de hilos. Por otra parte, se realizó un análisis profundo de la infraestructura actual de procesamiento de datos en la empresa, tanto su arquitectura como su base de datos.

De igual forma, este estudio recopiló datos relevantes sobre el rendimiento, la escalabilidad y la eficiencia del proceso. Los resultados obtenidos fueron validados por la aplicación de las normas ISO 25000 con enfoque en la eficiencia de desempeño y fiabilidad, específicamente el apartado 25010.

Tomando en cuenta las consideraciones presentadas anteriormente se recomienda:

- Evaluar si el servidor tiene la capacidad para albergar hilos es crucial. La orientación de la evaluación debe centrarse en la capacidad de procesamiento y la memoria. Es

esencial asegurarse de que, a pesar de la implementación de hilos, el servidor cuente con los recursos necesarios para evitar ralentizaciones.

- Poner especial atención a la estructuración de las peticiones. Es esencial diseñar y optimizar las peticiones de manera eficiente para garantizar un rendimiento óptimo del proceso ETL en Big Data. La correcta estructuración contribuirá significativamente a un funcionamiento fluido del sistema.
- Considerar el sistema operativo al ejecutar hilos, ya que es indispensable adaptarse a las configuraciones específicas del sistema operativo para asegurar una ejecución eficiente.
- Las sugerencias y buenas prácticas proporcionadas por la librería Concurrent.Futures. La modularización del código facilita la gestión de hilos, y el uso de un bloque main adecuado asegura el correcto funcionamiento de los hilos. Cumplir con las recomendaciones de la librería contribuirá de manera significativa a una implementación correcta del proceso ETL.

Referencias

Abukwaik, H., Gogolev, A., Groß, C., & Aleksy, M. (2020, Septiembre). OPC UA Realization for simplified commissioning of adaptive sensing applications for the 5G IIoT. Internet of things, 11. doi:https://doi.org/10.1016/j.iot.2020.100221

Ali, S. M. (2018, Marzo). Next-generation ETL Framework to address the challenges Posed by Big Data in DOLAP.

Baijens, J., Helms, R., & Iren, D. (2020). Applying scrum in data science projects. IEEE 22nd conference on business informatics (CBI), 1, pp. 30-38. Antwerp. doi:https://doi.org/10.1109/CBI49978.2020.00011

blogspot. (2020, abril 12). Retrieved from https://iso25000quality.blogspot.com/

Bourany, T. (2018). Las 5V de big data. In R. M. economía, Da tus datos, "Big data", economía y sociedad (pp. 27-31). La Découverte. Retrieved from https://www.cairn.info/revue-regards-croises-sur-l-economie-2018-2-page-27.htm

Cheesman de Rueda, S. (2010). CONCEPTOS BÁSICOS EN INVESTIGACIÓN. Retrieved Noviembre 2023, 06, from https://investigar1.files.wordpress.com/2010/05/conceptos.pdf

Coelho, F. (2023). Enciclopedia Significados. Retrieved Noviembre 06, 2023, from https://www.significados.com/investigacion/

Dalcin, L., Paz, R., Kle, P., & Cosimo, A. (2011, Septiembre). Parallel distributed computing using Python. Advances in Water Resource. Advances in Water Resources, 34(9), 1124-1139. doi:https://doi.org/10.1016/j.advwatres.2011.04.013

De DocuSign, C. (2021, Abril 15). Optimización del tiempo: 5 recursos que pueden

ayudarte. Retrieved from DocuSign: https://www.docusign.com/es-mx/blog/optimizacion-del-tiempo

Desempeño, E. d. (n.d.). Portal ISO 25000. Retrieved from https://iso25000.com/index.php/normas-iso-25000/iso-25010/21-eficiencia-de-desempeno

Fahimeh , R., Jie, L., & Farookh, H. (2013). Task Scheduling Optimization in Cloud Computing. In Service-Oriented Computing: 11th International Conference, ICSOC 2013, 8274, pp. 237-251. Berlin. Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-45005-1_17

Falco, M., & Robiolo, G. (2021). Building a Catalogue of ISO/IEC 25010 Quality Measures Applied in an Industrial Context. Journal of Physics: Conference Series, 1828(1), 1742-6596. doi:10.1088/1742-6596/1828/1/012077

IBM. (2023, Julio 13). IBM documentation. Retrieved from https://www.ibm.com/docs/es/xl-fortran-aix/16.1.0?topic=optimization-programming

Iso 25000. (n.d.). Retrieved from https://iso25000.com/index.php/normas-iso-25000/iso-25010

Kadenic, M. D., Koumaditis, K., & Junker-Jensen, L. (2023). Mastering scrum with a focus on team maturity and key components of scrum. Information and Software Technology, 153. doi:https://doi.org/10.1016/j.infsof.2022.107079

Luengo, C. (2023, Septiembre 28). Beneficios del uso de software de optimización de lineales. Retrieved from El Dínamo: https://www.eldinamo.cl/presentado-por/2023/09/25/beneficios-del-uso-de-software-de-optimizacion-de-lineales/

Mahmood, W., Usmani, N., Ali, M., & Farooqui, S. (2020). Benefits to Organizations after migrating to Scrum.

MariaDB. (n.d.). MyISAM Overview. Retrieved from https://mariadb.com/kb/en/myisam-overview/

Perdomo, W., & Zapata, C. M. (2021). Software quality measures and their relationship with the states of the software system alpha. Revista Chilena De Ingeniería, 29(2), 346–363. Retrieved from https://doi.org/10.4067/s0718-33052021000200346

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS), (pp. 42-47).

Schramme, M., & Macías, J. A. (2019). Analysis and measurement of internal usability metrics through code annotations. Software Quality Journal, 27(4), 1505-1530. Retrieved from https://link.springer.com/article/10.1007/s11219-019-09455-4

Seenivasan, D. (2023). ETL (Extract, Transform, Load) Best Practices . International Journal of Computer Trends and Technology, Volume 71 Issue 1, 40-44.

Seenivasan, D. (2023). Improving the Performance of the ETL Jobs. International Journal of Computer Trends and Technology, 7(3), 27-33. doi:https://doi.org/10.14445/22312803/IJCTT-V71I3P105

Sodian, L., Wen, J. P., Davidson, L., & Loskot, P. (2022). Concurrency and Parallelism in Speeding Up I/O and CPU-Bound Tasks in Python 3.10. 2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), (pp. 560-564). doi:10.1109/CEI57409.2022.9950068

Sucipto, S., Resti, N., Andriyanto, T., Karaman, J., & Qamaria, R. (2019). Transactional

database design information system web-based tracer study integrated telegram bot. In Journal of Physics: Conference Series, 1381. doi:10.1088/1742-6596/1381/1/012008

TDX. (2010). DEFINICIÓN DE LA INVESTIGACIÓN. Universidad de Lima. Retrieved Noviembre 07, 2023, from https://tdx.cat/bitstream/handle/10803/8895/12btAmbitometodologico.pdf?sequence=15

Tu, T., Liu, X., Song, L., & Zhang, Y. (2019). Understanding Real-World Concurrency Bugs in Go. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, (pp. 865-878). Retrieved from https://dl.acm.org/doi/abs/10.1145/3297858.3304069

Universidad de Lima. (2023). Citas y referencias en APA: tutorial. Retrieved from https://libguides.ulima.edu.pe/citas_referencias/Conceptos

Visual Paradigm. (n.d.). Visual Paradigm. Retrieved from Scrum Resources: Guide, Examples, Tutorial and more: https://www.visual-paradigm.com/scrum/

Yan, M., Lei, D., Xing, H., Ling, L., Yujing, F., Xiaochun, Y., . . . Yuan, X. (2020). HyGCN: A GCN Accelerator with Hybrid Architecture. In 2020 IEEE International Symposium on High Performance Computer Architecture, (pp. 15-29). Retrieved from https://ieeexplore.ieee.org/abstract/document/9065592

Yuhua, N., Miao , M., & Limei. (2011). Design and implementation of the business layer of the Tourism Business Information Collection and Distribution System based on the proxy and the factory patterns. In 2011 3rd International Conference on Computer Research and Development, 1, pp. 199-202. Shanghai.

doi:10.1109/ICCRD.2011.5764003

APÉNDICES